國立臺灣師範大學
資訊工程研究所碩士論文

指導教授：林順喜 博士

六子棋之棋型分類及審局函數之研究

On the Study of Pattern Classification and Evaluation Function for Connect6 Games

研究生：陳志宏 撰

中華民國一百年六月

摘要

六子棋是吳毅成教授所提出的一系列 K 子棋當中的一種，又稱連六棋（Connect6），是具備「規則簡單」、「變化複雜」、「遊戲公平」等特性的棋類遊戲。本研究以賴昱臣設計的六子棋程式 Ant，作為基礎程式進行改良。經過棋型分類、審局函數與迫著搜尋系統的修改，使得勝率及和率明顯提升、敗率明顯降低。實驗顯示，本研究方法確實提升 Ant 程式的棋力。

關鍵字：六子棋、人工智慧、迫著搜尋

ABSTRACT

Connect6 is a kind of k-in-a-row game that was introduced by Professor I-Chen Wu. It is a fair and complex game with simple rules. This study is based on a Connect6 program Ant developed by Yu-Chen Lai. We improve its pattern recognition, evaluation method, and threat-space search algorithm to promote its win rate significantly. Experimental results show these ideas really make Ant become much stronger.

Keyword: Connect6, Artificial Intelligence, Threat-Space Search.

目錄

表目錄

圖目錄

第1章　緒論

1.1 研究背景

六子棋於第十一屆奧林匹亞電腦賽局競賽（The 11th Computer Olympiad, 2006），正式加入成為比賽的項目之一，發展至今，已有愈來愈多同好加入六子棋的研究行列，各種推陳出新的行棋策略，以及日新月異的搜尋算法不斷發展，使得六子棋程式的棋力迅速提升，也讓參加奧林匹亞電腦賽局競賽─六子棋項目的隊伍，有逐年增加的趨勢，其中不乏高段棋力的六子棋程式。

六子棋係國立交通大學資訊工程系吳毅成教授發展出一系列 K 子棋的其中一種，又稱連六棋（Connect6），它改良自五子棋，卻不需要額外的禁手及開局規則，是具備「規則簡單」、「變化複雜」、「遊戲公平」等特性的棋類遊戲[3]。

首先介紹規則簡單的特性：六子棋屬雙人對弈棋類，對局時一般使用大小為 19×19 的棋盤，先手持黑、後手持白，除黑方第一手只能下一子，往後白黑雙方輪流各下兩子。在雙方對弈過程中，只要其中一方連續六子連成直線（往後稱連六），即為獲勝；若雙方持續無法形成連六直到盤面下滿，則稱雙方和棋。圖 1-1 為六子棋對弈之前三手示範。圖 1-2 展示一盤棋之對弈結果。

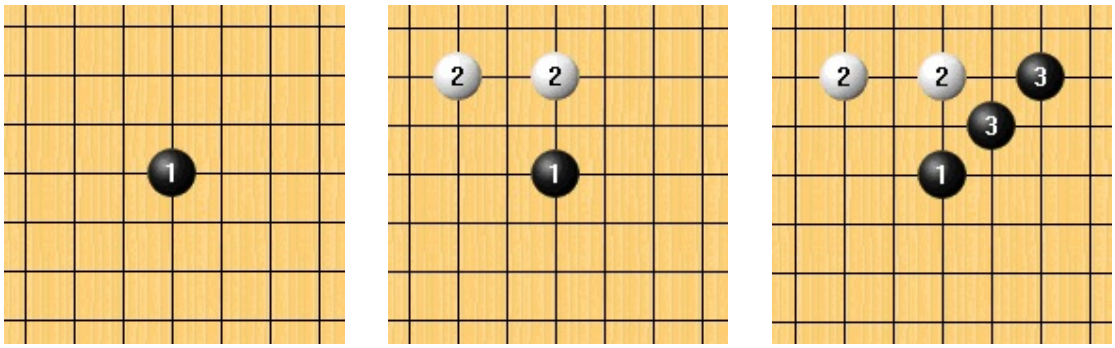


圖 1-1　由左至右分別為六子棋對弈之第一手、第二手、及第三手示範

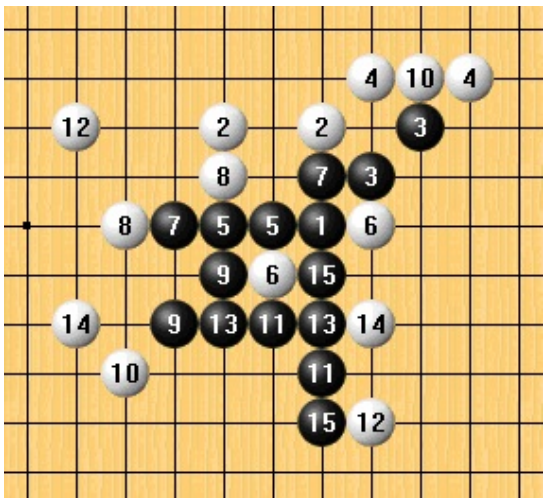


圖 1-2　黑方連六，獲得勝利

接著說明變化複雜的特性：六子棋看似規則簡單，但除第一手外，其餘每一手皆可下兩子，這意味著每一手之走步組合高達C_2^n種（n 表示棋盤上之空格數），而下滿整個棋盤共需要 181 手。由此可見六子棋遊戲變化萬千、極為複雜，其 state space 複雜度高達 10^{172}，game tree 複雜度亦達 10^{140}（以下 30 手計算），表 1-1 列出不同棋類的複雜度作為比較[2][7]。

最後說明遊戲公平之特性：以五子棋為例，雙方每手只能下一子，故黑方每下一子後，盤面上黑方的子數必多出白方一子；而白方每下一子後，盤面上白方的子數才等於黑方之子數，顯然優勢一直存在於黑方。相對於五子棋，六子棋在對弈過程中，某一方每下兩子後，盤面上其子數必多出對方一子，這也達到優勢互換的特性，直覺上也較為公平。目前尚無理論證明六子棋為明確不公平、單調不公平與經驗上不公平的遊戲，依定義六子棋仍屬於潛在公平性的遊戲[8]。

表 1-1　不同棋類遊戲之複雜度比較

Game	State space complexity	Game tree complexity
Awari	10^{12}	10^{32}
Checkers	10^{21}	10^{31}
Chess	10^{46}	10^{123}
Chinese Chess	10^{48}	10^{150}
Connect-Four	10^{14}	10^{21}
Dakon-6	10^{15}	10^{33}
Domineering (8×8)	10^{15}	10^{27}
Draughts	10^{30}	10^{54}
Go (19×19)	10^{172}	10^{360}
Go-Moku (15×15)	10^{105}	10^{70}
Hex (11×11)	10^{57}	10^{98}
Kalah (6,4)	10^{13}	10^{18}
Nine Men’s Morris	10^{10}	10^{50}
Othello	10^{28}	10^{58}
Pentominoes	10^{12}	10^{18}
Qubic	10^{30}	10^{34}
Renju (15×15)	10^{105}	10^{70}
Shogi	10^{71}	10^{226}
Connect6	10^{172}	10^{140}

1.2 研究動機及目的

奧林匹亞電腦賽局競賽每年都吸引世界各地頂尖的電腦棋類程式參與，而六子棋也是此競賽的比賽項目之一，該如何設計出能與世界列強比擬的六子棋程式，並能從中脫穎而出，就變得十分具有挑戰性。

本研究基於改良六子棋程式之設計與提升程式棋力，藉由國立台灣師範大學資訊工程系林順喜教授所指導之研究生賴昱臣設計的六子棋程式 Ant，作為基礎

程式進行改良。該程式於第十五屆奧林匹亞電腦賽局競賽（The 15th Computer Olympiad, 2010）首次登場，而本人很榮幸成為 Ant 程式的操作者。在該競賽中，國立東華大學的 Kavalan、北京理工大學的 BITconnect6 與 Ant，名列前茅的三支隊伍在正規賽中不分軒輊，直到延長加賽 Ant 不幸敗給 Kavalan，獲得第 4 名。表 1-2 為第十五屆奧林匹亞電腦賽局競賽—六子棋項目的比賽結果[5]。

表 1-2　第十五屆奧林匹亞電腦賽局競賽—六子棋項目之比賽結果

Program Name	**Origin**	**Games**	**Score**	**Title**
MoreThenFive	China	5	10	1 (Gold)
Kavalan	Taiwan	7	9	2 (Silver)
BITconnect6	China	7	8	3 (Bronze)
Ant	Taiwan	7	7	4
Crazy6	Japan	5	2	5
MakeIt6	The Netherlands	5	0	6

經過第十五屆奧林匹亞電腦賽局競賽的洗禮，本人對六子棋逐漸產生興趣，於是將 Ant 的棋型分數做一調整，並參與 TAAI 2010 電腦對局比賽，在此次比賽很幸運地擊敗國立東華大學的 Kavalan，得到第二名的殊榮，這也說明棋型分類確實在六子棋佔有一席之地。表 1-3 為 TAAI 2010 電腦對局比賽的結果[6]，獎牌照片收錄於附錄 A。

表 1-3　TAAI 2010 電腦對局比賽—六子棋項目比賽結果

Program Name	**Origin**	**Authors**	**Medal**
SIMC	Taiwan	Ping-Hung Lin, Hsin-Ti Tsai, Hao-Hua Kang, I-Chen Wu	Gold
Ant	Taiwan	Yu-Chen Lai, Chih-Hung Chen	Silver
Kavalan	Taiwan	Jung-Kuei Yang, Shi-Jim Yen	Bronze

雖在 TAAI 2010 電腦對局比賽獲得銀牌的榮耀，但仍與國立交通大學的

SIMC 程式有明顯落差，而其中一個原因即是審局函數不夠精準。審局函數要好，棋型必然要先妥善分類；若棋型分類雜亂無章，則再巧妙的審局函數都回天乏術，更別提其他有效的搜尋算法。故本篇論文著重在棋型的分類及審局函數的研究，期望改善審局函數不夠精準的問題，進而提升六子棋程式─Ant 之棋力。

1.3 論文架構

本論文分為五個章節，第一章介紹六子棋的背景知識、研究動機及目的，第二章探討六子棋之相關文獻，第三章說明研究方法，緊接著第四章報告研究進度，最後，第五章敘述預期得到之結果。整體架構如圖 1-3 所示。

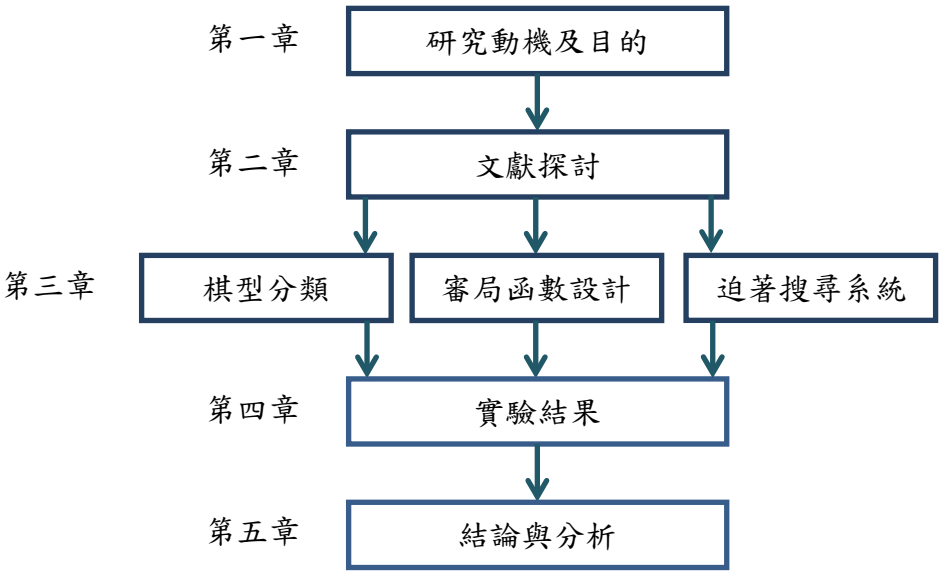


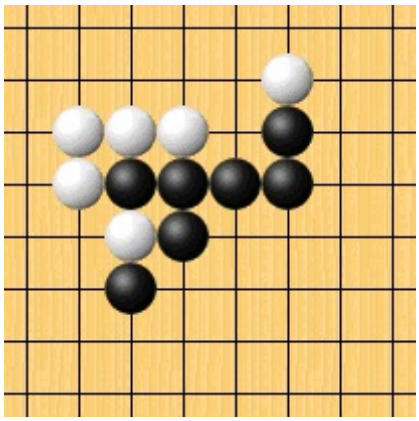
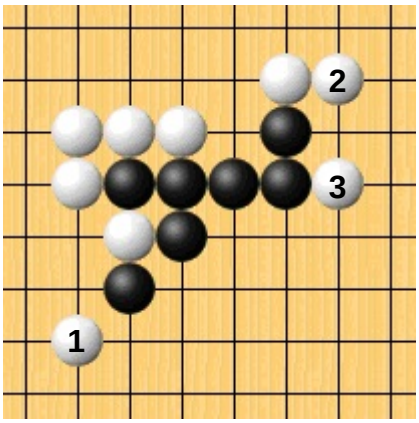
圖 1-3　論文架構圖

第2章　文獻探討

2.1 遊戲策略

敘述六子棋攻擊與防守的策略前，需要先了解迫著（threat）的意義；就六子棋而言，迫著定義如下：若白方至少需要下 t 子來阻止黑方連六，則黑方擁有 t 迫著；反之亦然[7]。簡單來說，若黑方擁有一個迫著，則白方至少需要下一子來阻止黑方形成連六；若黑方擁有兩個迫著，則白方至少需要下兩子來阻止黑方形成連六。表 2-1 列出黑方形成單迫著、雙迫著、與三迫著的範例及白方可能的阻擋方法。

表 2-1　黑方形成單迫著、雙迫著、與三迫著的範例及白方可能的擋法

迫著數	黑方形成 t 迫著之範例	白方至少需要下 t 子之可能擋法
單迫著		
雙迫著		
三迫著		

對迫著的概念有基本認識後，就可開始討論攻擊與防守應當注意那些要領。由於六子棋每手最多只能下兩子，若一方擁有三迫著或三迫著以上，則下一手該方必定獲勝。故在部署攻擊策略時，我方須思考如何形成三個或三個以上之迫著；

而在守防策略時，我方須遏止對手形成三個或三個以上之迫著。只要能將攻守策略妥善運用，勢必能使棋力大大提升。

2.2 迫著搜尋

於 2.1 節提及迫著的概念後，本節介紹一個與迫著相關的搜尋算法─迫著搜尋（threat-space search），迫著搜尋的想法為攻擊方存在一連續形成迫著的進攻走步，若防守方無法抵擋攻擊方一連串的迫著攻擊，則攻擊方可依迫著搜尋的路徑進攻並獲得勝利[1]。圖 2-1 為迫著搜尋示意圖。

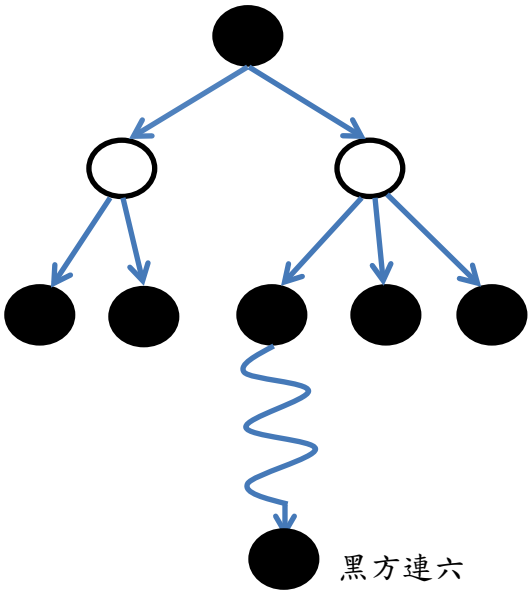


圖 2-1　黑方連六，迫著搜尋成功

目前蒐集到兩篇關於六子棋的論文，皆有說明其搜尋演算法、棋型種類與審局函數，此兩篇論文分別為“結合單迫著與雙迫著搜尋之六子棋程式之研發[10]”、“電腦六子棋程式 X6 之設計與實作[9]”。其中，第一篇論文所開發的程式名稱為 Ant，第二篇論文設計的程式名稱為 X6。本節先介紹這兩支程式之搜尋演算法，後續章節再介紹兩支程式的棋型種類與審局函數。Ant 程式在搜尋方

面，係結合單、雙迫著搜尋搭配使用；X6 程式則單純使用雙迫著搜尋。

2.3 棋型介紹

在進入六子棋棋型介紹之前，本節先說明由迫著理論所衍生出之棋型的死型與活型。死型定義為：若棋型 A 只需要再下（*4-k*）子即可形成單迫著，則稱 A 為 *dead-k* 棋型（*死-k*）；活型定義為：若棋型 B 只需再下（*4-k*）子即可形成雙迫著，則稱 B 為 *live-k* 棋型（*活-k*）。表 2-2 列出死二、死三、活二與活三的範例。

表 2-2　死二、死三、活二與活三範例

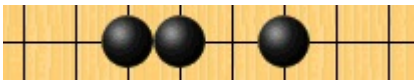
	黑方棋型	**黑方下（*4-k*）子可形成迫著**
死二		
死三		
活二		
活三		

本節將介紹六子棋程式 Ant 與 X6 之棋型種類及分數，並於下一節接著說明六子棋程式 Ant 與 X6 之審局函數。

2.3.1　六子棋程式 Ant 之棋型種類

由 Theo van der Storm[4]所開發的六子棋程式─MeinStein 於 2008 年開放了原始碼，而六子棋程式 Ant 最初之棋型亦是沿用 MeinStein 的棋型種類。棋型種類有 10 種，分別為死二、死三、活二、活二死三、活三、死四、死五、活四、活四死五、以及活五。表 2-3 展示此 10 種棋型之圖示、分數、及迫著數。

表 2-3　六子棋程式—Ant 之棋型種類、圖示、分數、及迫著數

棋型種類	棋型圖示	棋型分數	迫著數
死二		100	0
死三		300	0
活二		400	0
活二死三		600	0
活三		1000	0
死四		2000	1
死五		2000	1
活四		4000	2
活四死五		4500	2
活五		4500	2

2.3.2　六子棋程式 X6 之棋型種類

X6 為東華大學所研發之六子棋程式，其棋型主要分四大型態：連續出現我方棋子稱活型、我方兩群棋子間隔一格稱眠型、我方兩群棋子間隔二格（或我方兩群棋子間隔一格且一邊出現對方棋子）稱斷型、連續出現我方棋子且一邊出現對方棋子稱死型。每一型態再依二個棋子、三個棋子、四個棋子、及五個棋子做區分，共有 16 種（4 大型態×4 種棋子個數＝16）。另外，我方連六為第 17 種棋型。表 2-4 列出此 17 種棋型之圖示、及棋型分數。

表 2-4　六子棋程式—X6 之棋型種類、圖示、及分數

棋型種類	棋型圖示	我方棋型分	對方棋型分
死二		0	0
斷二		0	0
眠二		2	4
活二		3	6
死三		0	3
斷三		0	5
眠三		5	9
活三		7	13
死四		2	5
斷四		1	6
眠四		3	13
活四		4	24
死五		306	106
斷五		300	100
眠五		302	102
活五		330	130
連六		900	200

上述六子棋程式 Ant 與 X6 之部分棋型並無理論依據，導致有定義不明確的

疑慮，容易造成玩家混淆，例如 Ant 程式的活二死三棋型等同於 X6 的眠三。本論文的第三章將針對棋型分類做進一步的探討。

2.4 審局函數

審局函數主要用於計算某個空點的分數（此空點的分數爾後稱為子力），作為判斷該位置優劣的依據。在這一節繼續介紹六子棋的審局函數，首先說明 Ant 程式之計算子力的方式，接著介紹 X6 程式如何審局。

2.4.1 六子棋程式 Ant 之審局函數

Ant 程式在評估某一個空點之優劣時，其子力主要由下列四個分數所組成：該空點下了我方子的棋型分數−下子前我方之棋型分數（ΔP_1）、該空點下了對方子的棋型分數−下子前對方之棋型分數（ΔP_2）、該空點下了我方子的迫著數−下子前我方之迫著數（ΔT_1）、該空點下了對方子的迫著數−下子前對方之迫著數（ΔT_2）。另外，依照是否使用迫著搜尋而有不同的子力計算公式，下列將一一說明：

I. 沒有使用迫著搜尋時之子力公式=$\Delta P_1 + \Delta P_2$。以圖 2-2 為例，黑方欲求位置 A 之子力，其 ΔP_1 = (4500[表 2-3 之活四死五] – 2000 [死四]) + (400[活二] – 0) = 2900、ΔP_2 = (400[活二] – 0) + (400[活二] – 0) = 800，故可得知位置 A 下黑子的子力= 2900 + 800 = 3700。

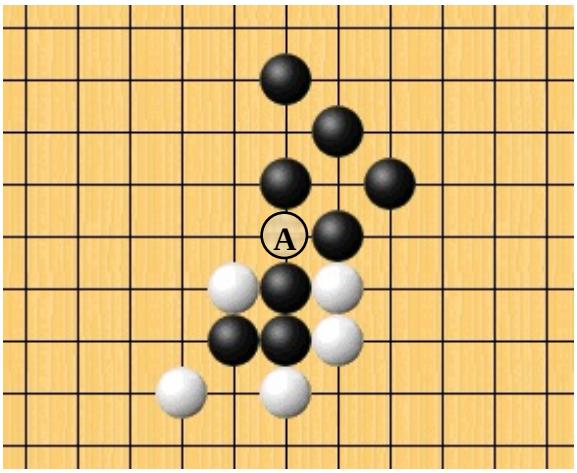


圖 2-2　無迫著搜尋時，黑方欲求位置 A 之子力

II.　使用迫著搜尋時，攻擊方之子力公式=$\Delta P_1 + 10000 \times \Delta T_1$。以圖 2-3 為例，黑方欲求位置 B 之子力，其 $\Delta P_1 = (4000_{[活四]} - 1000_{[活三]}) = 3000$、$\Delta T_1 = (2_{[活四]} - 0_{[活三]}) = 2$，故可得知位置 B 下黑子的子力= $3000 + 10000 \times 2 = 23000$。

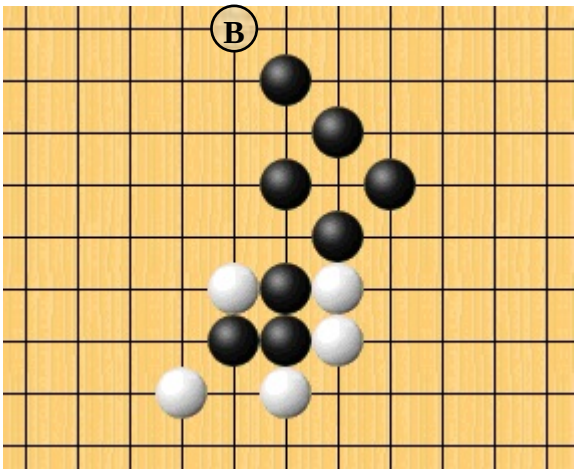


圖 2-3　使用迫著搜尋時，黑方欲求位置 B 之子力

III.　使用迫著搜尋時，防守方之子力公式= $2000 \times \Delta T_1 + \Delta P_2 - 10000 \times \Delta T_2$。以圖 2-4 為例，白方欲求位置 C 之子力，其 $\Delta T_1 = (0 - 0) = 0$、$\Delta P_2 = (2000_{[死五]} - 2000_{[死四]}) = 0$、$\Delta T_2 = (0 - 1_{[死四]}) = -1$，故位置 C 下白子

的子力= 2000 × 0 + 0 – 10000 × (-1) = 10000。

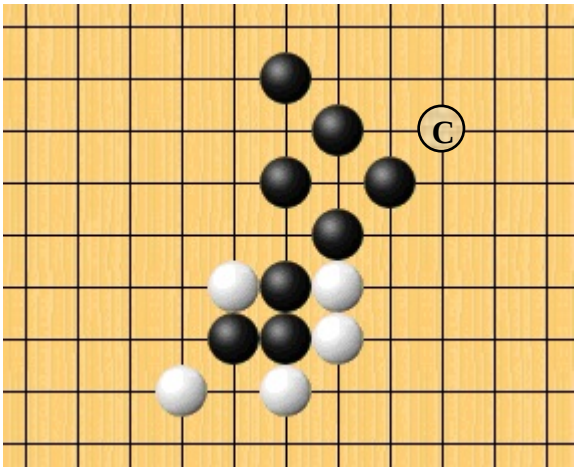


圖 2-4　使用迫著搜尋時，白方欲求位置 C 之子力

IV. 使用迫著搜尋時，防守方無視遊戲規則，下了第三子之子力公式= ΔP_1 + ΔP_2。以圖 2-5 為例，白方欲求位置 D 之子力，其 ΔP_1 = (100[死二] – 0) = 100、ΔP_2 = (0 – 0) = 0，故位置 D 下白子的子力= 100 + 0 = 100。

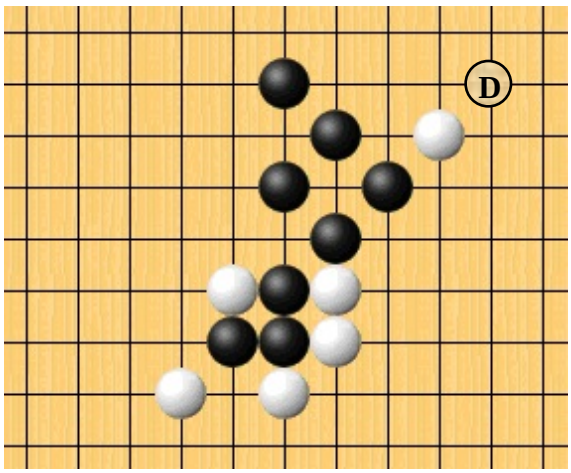


圖 2-5　使用迫著搜尋時，白方欲求位置 D 之子力

2.4.2　六子棋程式 X6 之審局函數

六子棋程式 X6 在審局時，會根據六種不同的情況對子力作調整，下列針對各種加權情況做簡單的介紹。

I. 必勝棋型組合：某位置下子後造成迫著數 ≥ 3，加權最多。

II. 優勢攻擊棋型組合：下了某位置後形成迫著及兩個活二（或兩個眠二，參考表 2-4 之棋型種類），亦會加權。

III. 迫著位置：於某位置下子能產生單迫著，但在另一位置下子卻能形成雙迫著，此種情況會針對形成迫著數的多寡而有不同的加權。

IV. 高子力與高度活動空間：若子力及活動空間> 2，則子力加權 43%；若子力及活動空間> 1，則子力加權 33%；若子力及活動空間> 0，則子力加權 25%。

V. 開局時的特殊走法：X6 程式內建六種開局走法，每種走法之加權分數亦不同。

VI. 攻擊或防禦的加權：進攻時，我方棋型分數=我方的棋型分數+對方的棋型分數，請參考表 2-4；防守時，我方棋型分數維持不變，不再累加對手的分數。

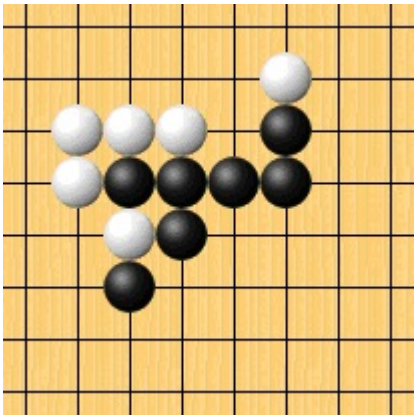
第3章　研究方法

本研究以賴昱臣所研發的六子棋程式Ant作為基礎(此版本往後稱Ant_1.4)，針對棋型分類、審局函數、候選步的篩選、時間控管、迫著搜尋系統等，加以修改。本章將分五節各自說明每一部分之想法。

3.1 棋型分類

於棋型分類上，Ant_1.4 版本與東華大學所研發的 X6 程式存在棋型定義不明確的疑慮。為避免類似的問題，本研究以迫著數作為棋型分類的準則。另外，三迫著或三迫著以上之棋型，皆確保下一手必獲勝，故把三迫著以上之棋型歸類為三迫著。以上述分法將棋型重新分類成：死一、活一、死二、活二、死三、活三、單迫著、雙迫著、三迫著、連六。如此一來，每種棋型皆有理論依據，不會造成玩家混淆。表 3-1 列出這 10 種棋型之範例及給定的棋型分數。

表 3-1　以迫著數為分類規則之棋型範例

棋型種類	棋型圖示	棋型分數	迫著數
死一		5	0
活一		15	0
死二		100	0
活二		400	0
死三		300	0
活三		850	0
單迫著		600	1
雙迫著		1700	2
三迫著		2600	3
連六		999999	10

3.2 審局函數設計

於 2.4 節所提的兩個審局函數較為複雜，本研究採用另一種審局方法，其想法為考慮某空點對雙方的影響大小，作為評估位置優劣的方式。評估公式主要由下列四個分數所組成：該空點下我方子所得到的分數（ΔV_1）、該空點下對方子

所得到的分數（ΔV_2）、該空點下我方子的迫著數-下子前我方之迫著數（ΔT_1）、該空點下子前對方之迫著數（ΔT_2）。另外，審局函數再依是否使用迫著搜尋而分成兩種計算公式，下列將分別說明此兩種計算方法：

I. 無迫著搜尋時之子力公式= $\Delta V_1 + \Delta V_2$。以圖 3-1 為例，黑方欲選擇 E、F、G 三點中最佳的位置。其中位置 E 的子力= $\Delta V_1 + \Delta V_2 = \left(\sum_{四個方向} 黑方棋型分數\right) + \left(\sum_{四個方向} 白方棋型分數\right)$ = (850[表 3-1 之活三] + 100[死二] + 5[死一] + 5[死一]) + (0 + 5[死一] + 400[活二] + 400[活二]) = 1765。位置 F 的子力= (5[死一] + 5[死一] + 15[活一] + 850[活三]) + (400[活二] + 100[死二] + 15[活一] + 5[死一]) = 1395。位置 G 的子力= (850[活三] + 5[死一] + 15[活一] + 5[死一]) + (5[死一] + 400[活二] + 15[活一] + 850[活三]) = 2145。故審局函數會挑選分數高的位置 G。

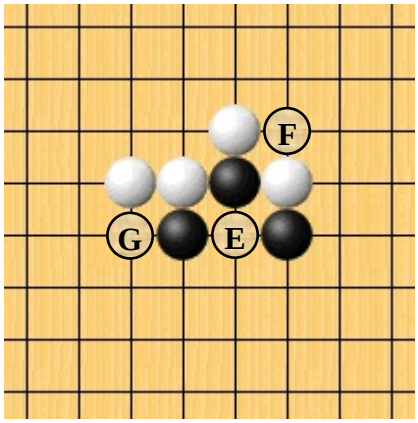


圖 3-1　無迫著搜尋時，黑方下哪個位置較佳

II. 有迫著搜尋時之子力公式= $\Delta V_1 + \Delta V_2 + 10000 \times \Delta T_1 + 100000 \times \Delta T_2$。以圖 3-2 為例，黑方欲選擇 H、I、J 三點中最佳的位置。位置 H 的子力= $\Delta V_1 + \Delta V_2 + 10000 \times \Delta T_1 + 100000 \times \Delta T_2 = \left(\sum_{四個方向} 黑方棋型分數\right) + \left(\sum_{四個方向} 白方棋型分數\right) + 10000 \times \left(\sum_{四個方向} 黑方下子後迫著數 -\right.$

$\text{下子前迫著數}\big) + 100000 \times \big(\sum_{\text{四個方向}} \text{白方 下子前迫著數}\big)$ = (15[活一] + 850[活三] + 5[死一] + 15[活一]) + (15[活一] + 5[死一] + 100[死二] + 15[活一]) + 10000 × {(0 – 0) + (0 – 0) + (0 – 0) + (0 – 0)} + 100000 × { 0 + 0 + 0 + 0 } = 1020。位置 I 的子力= (600[單迫著] + 15[活一] + 5[死一] + 400[活二]) + (5[死一] + 15[活一] + 400[活二] + 5[死一]) + 10000 × {(1 – 0) + (0 – 0) + (0 – 0) + (0 – 0)} + 100000 × { 0 + 0 + 0 + 0 } = 11445。位置 J 的子力= (15[活一] + 15[活一] + 15[活一] + 5[死一]) + (15[活一] + 15[活一] + 15[活一] + 600[單迫著]) + 10000 ×{(0 – 0) + (0 – 0) + (0 – 0) + (0 – 0)} + 100000 × { 0 + 0 + 0 + 1 } = 100695。故審局函數會挑選分數最高的位置 J。

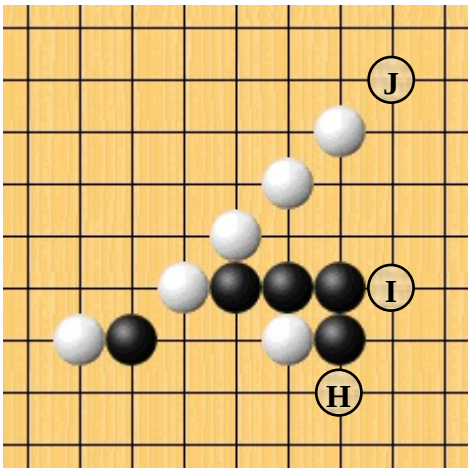


圖 3-2　有迫著搜尋時，黑方下哪個位置較佳

以此想法作為審局方式較為簡單，也不會偏重黑方或白方，能較公平地找出目前盤面對雙方皆重要的位置，並達到攻守並重的目的。至於 ΔT_2 權重較高的原因是為了阻擋對方形成的迫著，若 ΔT_1 與 ΔT_2 的權重一樣，則圖 3-2 的例子就會下位置 I，而不擋位置 J。

3.3 候選步的篩選

由於六子棋一次可下兩子，所以在篩選候選步時，須將棋盤上的所有空點兩兩配對，組成眾多候選步作為後續搜尋的依據，其中每組候選步由兩個空點所組成。搜尋時，各組候選步依分數高低排序，分數較高的候選步優先展開搜尋，故在有限的時間內，只能搜尋若干個分數較高的候選步，並無法展開所有的候選步。然而 Ant_1.4 版本在挑選候選步時，將所有分數大於等於死二之棋型分數[表 2-3]，皆拿來兩兩配對。以圖 3-3 為例，某空點下了黑子或白子後，分數大於等於 100 分的位置共有 125 個，也就是說 Ant_1.4 版本會產生 7750（C_2^{125}）種候選步組合，而後續搜尋卻只展開少數幾個候選步，換句話說，有大量的配對計算是多餘的，會造成程式額外的負擔。

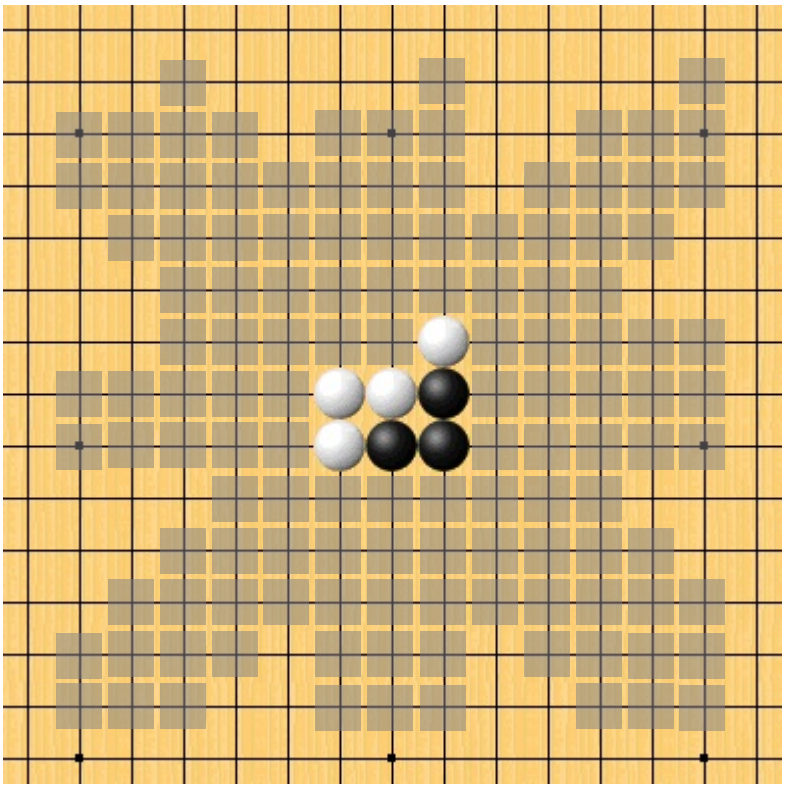


圖 3-3　Ant_1.4 版本的候選步區域

為減少候選步配對過多造成的浪費，本研究將配對標準提高，更改成某空點

下了黑子或白子後，分數大於等於死三之棋型分數[表 3-1]者，才符合配對的門檻。至於門檻訂為死三的理由，是因為死三棋型再與另一子配對後，可能形成單迫著棋型，而形成單迫著棋型才能進一步進入迫著搜尋階段。提高候選步配對標準後，重新檢視圖 3-3 的例子，發現程式只產生 1596（C_2^{57}）種候選步組合，足足節省 6154 個多餘計算。圖 3-4 為提高標準後的候選步區域。

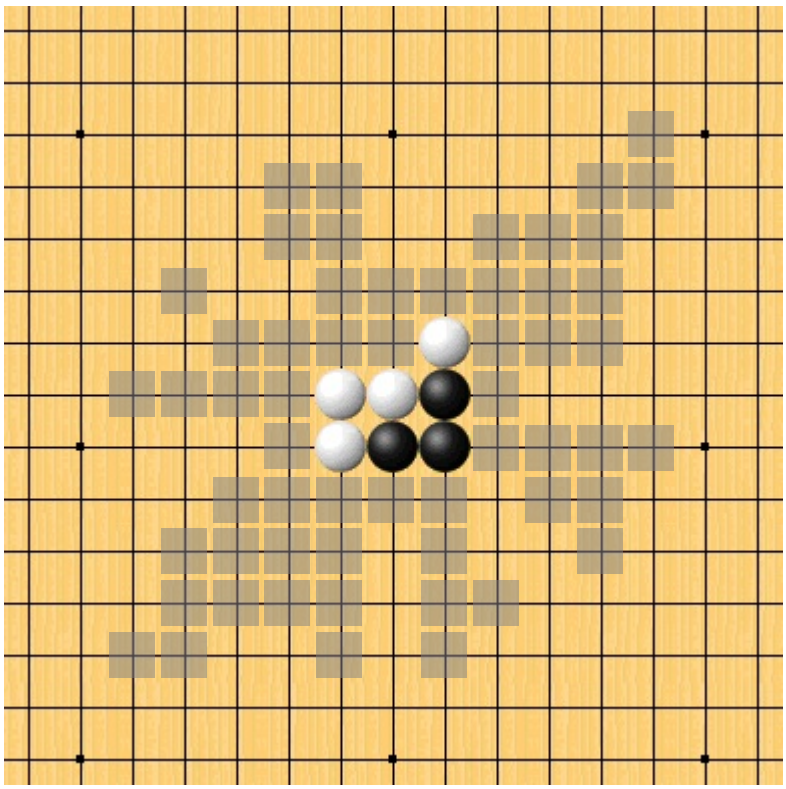


圖 3-4　提高標準後的候選步區域

3.4 時間控管

在時間控制部分， Ant_1.4 版本於下棋前，會先設定一盤棋的總思考時間，預設為 800 秒。下棋時，若尚有思考時間，則程式就能進入搜尋階段，尋找迫著路徑，直到 800 秒耗盡才不再進行搜尋。這個做法用於下棋時，經常開局幾手便把時間耗光，導致往後盤面都無時間搜尋，只能草草出手，在這種情況下要獲得

勝利實在非常困難，且容易陷入對手的陷阱當中。圖 3-5 為 Ant_1.4 版本的時間控制流程圖。

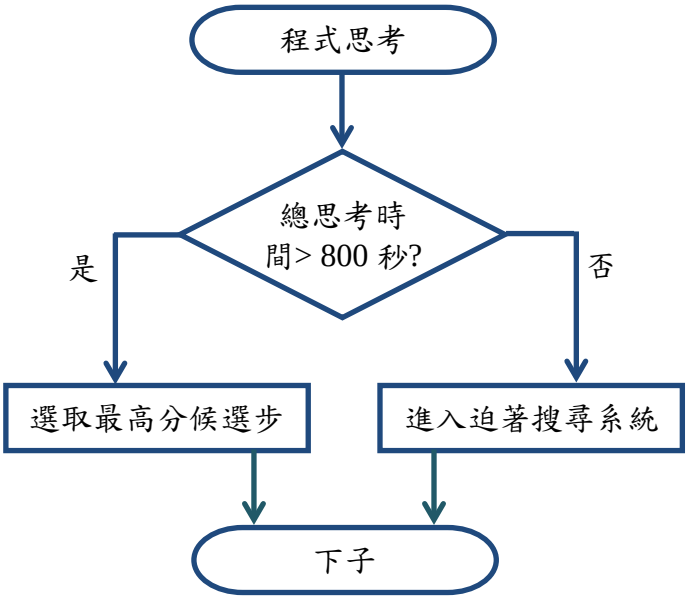


圖 3-5　Ant_1.4 版本之時間控制流程圖

上述做法風險較高，本研究採用另一種時間控管方法，想法為每一手先預設給若干秒的搜尋時間，在下棋時，第一手如果未將自己的搜尋時間用完，則第一手剩餘的時間會挪給第二手使用，此時第二手的搜尋時間=第二手預設秒數+第一手剩餘秒數。若第二手也未將時間用完，則再把剩下的時間轉移給第三手使用，往後幾手以此類推。如此一來，若前面幾手都未將時間用盡，則當下就有一段較長的時間可以運用，也可確保往後盤面仍有時間搜尋，以增加獲勝機會、降低踏入陷阱的風險。圖 3-6 為本研究所使用的時間控制流程圖。

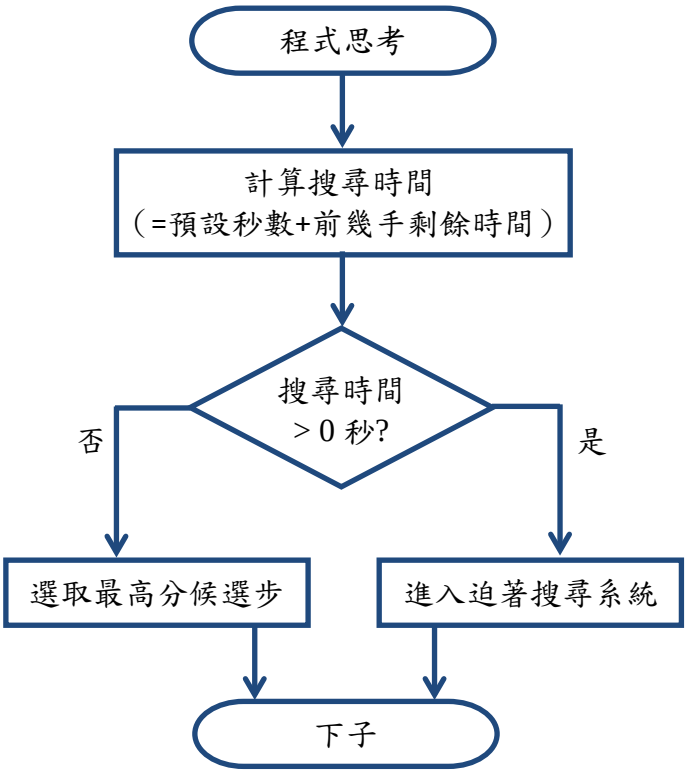


圖 3-6　本研究使用之時間控制流程圖

3.5 迫著搜尋系統

Ant_1.4 版本在迫著搜尋方面，採取單迫著與雙迫著混合使用，其迫著系統分成兩個階段，分別為我方迫著搜尋與檢查對方迫著搜尋。在程式進入搜尋系統時，第一階段針對我方候選步進行迫著搜尋，取出 6 個我方分數較高的候選步組合，依序對分數最高的候選步展開，尋找是否存在迫著成功的路徑。若迫著成功，則程式離開迫著系統，選擇此候選步的位置下棋；若迫著不成功，則換分數次高的候選步進行搜尋展開。直到 6 個候選步皆迫著失敗，才進入第二階段。

第二階段是檢查對方迫著搜尋是否成功，做法是針對我方 6 個分數較高的候選步組合，檢查我方下了最高分的候選步後，對方是否仍可以迫著成功。若對方迫著失敗，則程式離開迫著系統，選擇此候選步的位置下棋；若對方迫著成功，則換分數次高的候選步進行檢查。如果 6 個候選步皆讓對手迫著成功，程式就會

投降認輸，選擇第 6 個候選步的位置下棋。圖 3-7 為 Ant_1.4 版本的迫著搜尋流程。

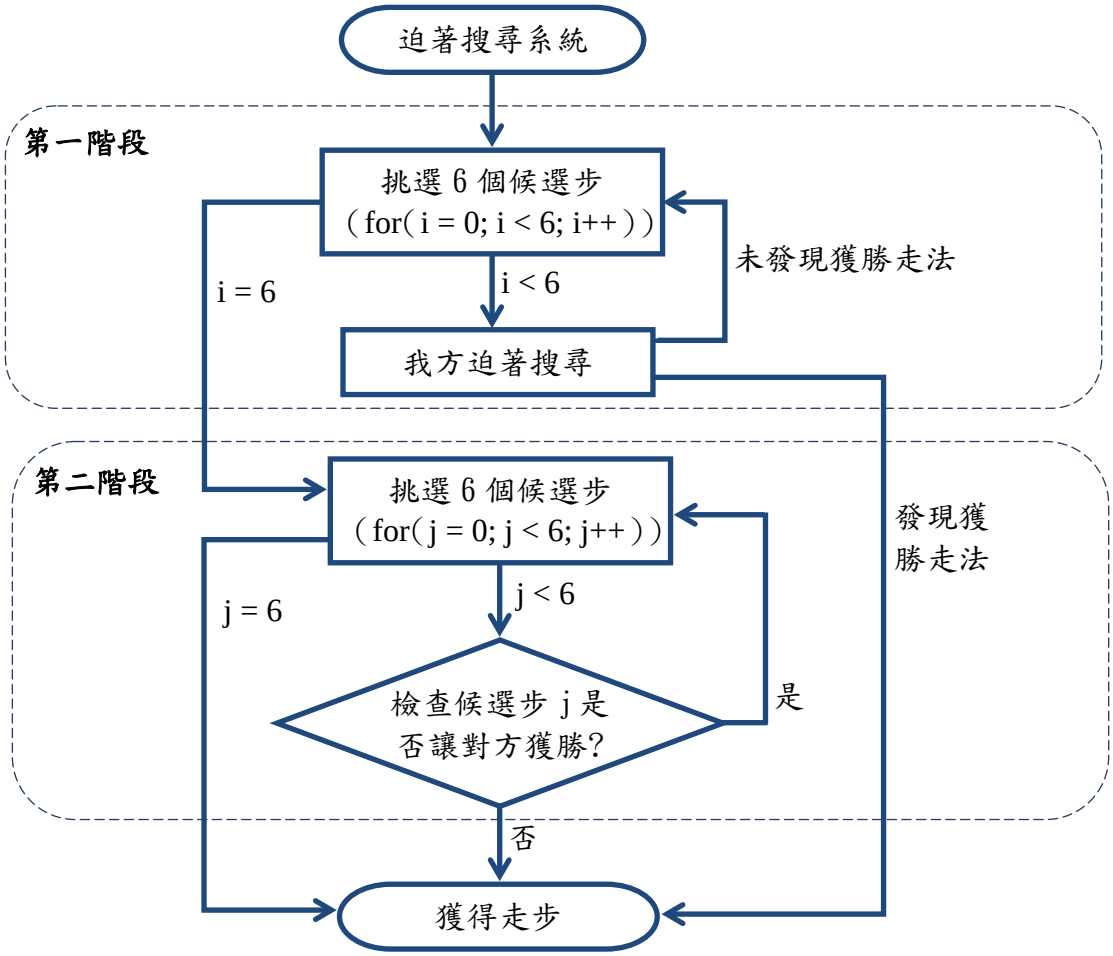


圖 3-7　Ant_1.4 版本之迫著搜尋流程圖

Ant_1.4 版本在第二階段的檢查，有偏重攻擊方的傾向，且其單迫著誤判率較高，本研究針對這些部分做改進，在迫著搜尋上，只選擇誤判風險較低的雙迫著搜尋，並將迫著搜尋系統分成三階段，第一階段維持不變，仍然是針對我方候選步進行迫著搜尋、第二階段為替對方候選步進行迫著搜尋、最後階段選擇我方分數最高的候選步。

更改後之迫著搜尋系統執行流程為：第一階段針對我方候選步進行迫著搜尋，程式預先取出 15 個我方分數較高的候選步，優先對分數最高的候選步展開，尋

找是否存在迫著獲勝的路徑。若迫著搜尋成功，則程式離開迫著搜尋系統，選擇此候選步的位置下棋;若迫著搜尋不成功,則換分數次高的候選步進行搜尋展開。直到 15 個候選步皆迫著搜尋失敗，才進入第二階段。

第二階段的想法是我方先不下子，接著讓對方進行迫著搜尋，做法與第一階段相同，差別在攻守互換，看對方是否能找出迫著成功的路徑。若對方迫著搜尋成功，則程式離開迫著搜尋系統，我方選擇對手會獲勝的下法，在此位置下棋；若對方迫著搜尋失敗，則換對方分數次高的候選步進行迫著搜尋。直到對方 15 個候選步皆迫著搜尋失敗，才進入第三階段。

進入到第三階段表示雙方迫著搜尋皆失敗，此時程式選擇我方分數最高的候選步，等待下一波的攻防戰。圖 3-8 為更改後之迫著搜尋流程圖。

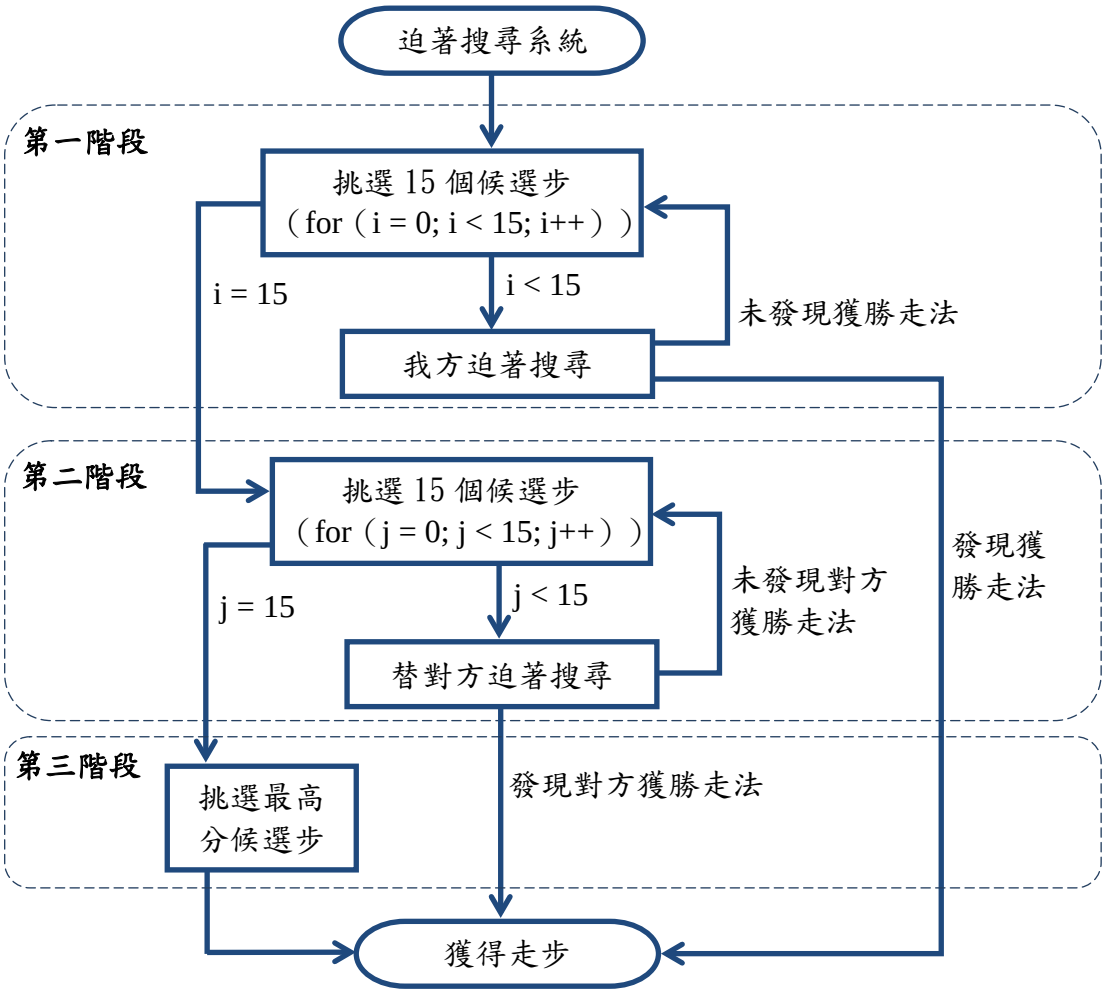


圖 3-8　本研究之迫著搜尋流程圖

第4章　實驗結果

本章開始先對程式介面與測試平台做描述，接著實驗第三章所提出的方法，這些方法修改出 Ant_1.6 與 Ant_1.7 兩個版本，為了比較修改後的效果，本章會針對未修改過的 Ant_1.4 版本（ICGA 2010 第四名），與 X6 程式（東華大學所研發，ICGA 2007 金牌）、NCTU6 程式（交通大學所研發，ICGA 2006 金牌）兩支程式對戰，統計其對戰結果做為日後比較的依據。接著再使用 Ant_1.6 與 Ant_1.7 兩個版本，分別與 Ant_1.4 版本、X6、NCTU6 測試，統計各版本的勝率、和率、敗率，再與 Ant_1.4 版本相互對照，以檢視本研究方法的成效好壞。

4.1 程式介面與測試平台介紹

本研究所開發之 Ant 程式係使用 Borland C++ builder 2010 做為開發工具，程式介面如圖 4-1 所示。介面右上方按鈕由左上至右下，依序為開始新遊戲、開啟棋譜、儲存棋譜、顯示執行資訊、移到初始盤面、後退一手、前進一手、移至最終盤面、搜尋時間、搜尋深度與程式思考等功能。程式思考後的走步會顯示在介面左方的棋盤上。

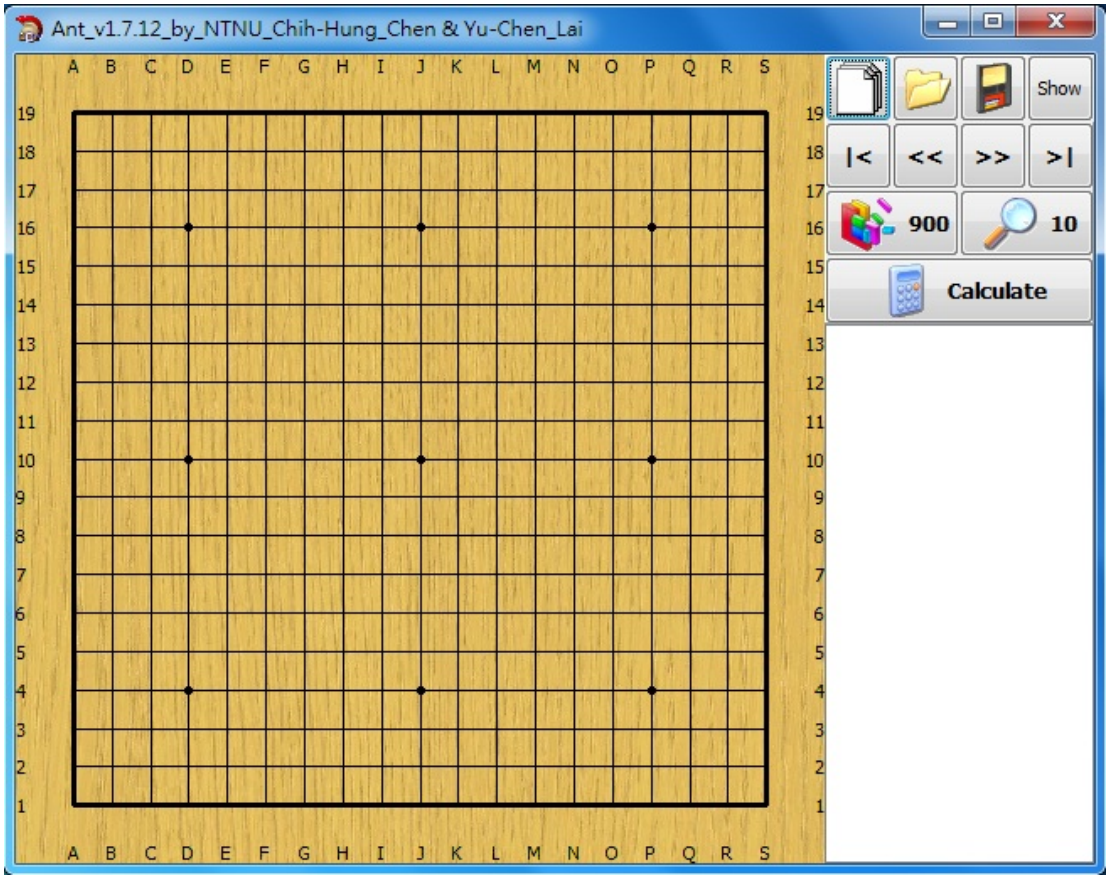


圖 4-1　Ant 程式之介面

實驗平台使用 Intel Core i3-M370 2.4GHz CPU、2G Ram、Windows 7 Enterprise 64 位元作業系統的筆記型電腦。測試對手參數設定如下：X6 程式 Rank 設定為 9、NCTU6 程式等級設定為 3，程式思考時間為 30 分鐘。由於 Ant 程式每次下的走步都相同，為了增加測試盤面的變化，所以棋局初始的前兩手皆由人工操作，第三手開始交由程式思考。圖 4-2 為 X6 程式的介面。圖 4-3 為 NCTU6 程式的介面。本章之測試盤面如圖 4-4 所示[10]。

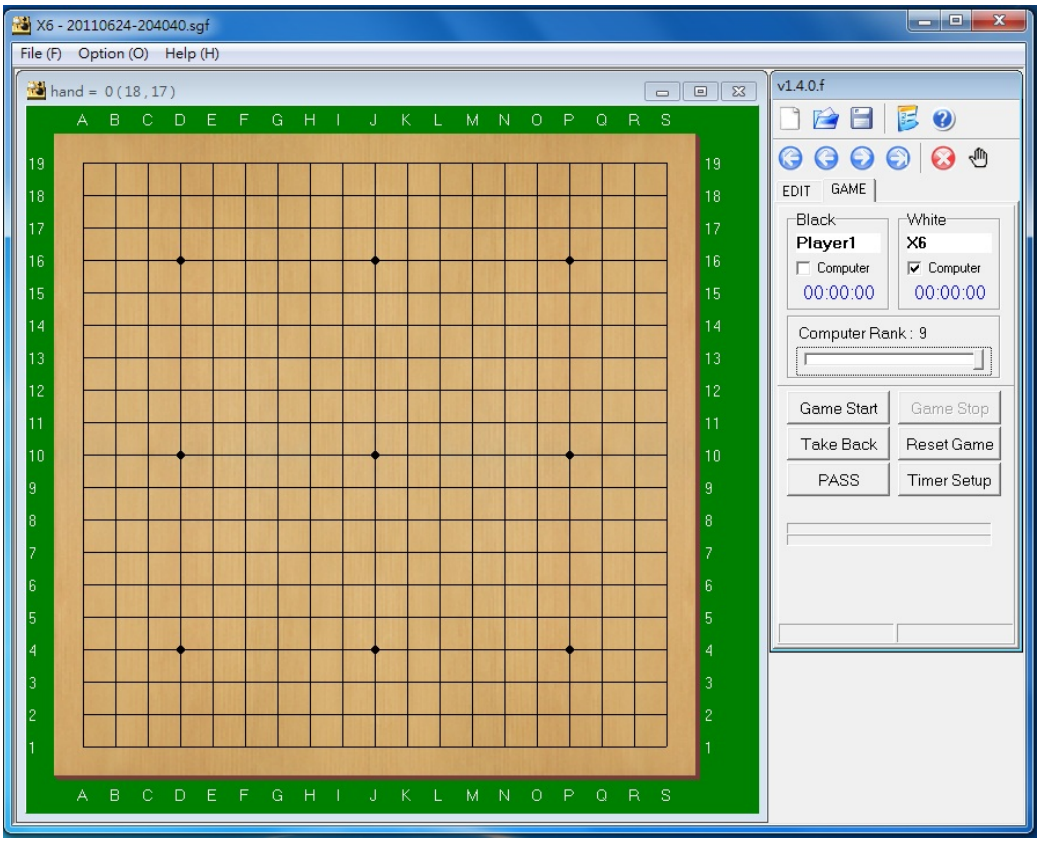


圖 4-2　X6 程式之介面

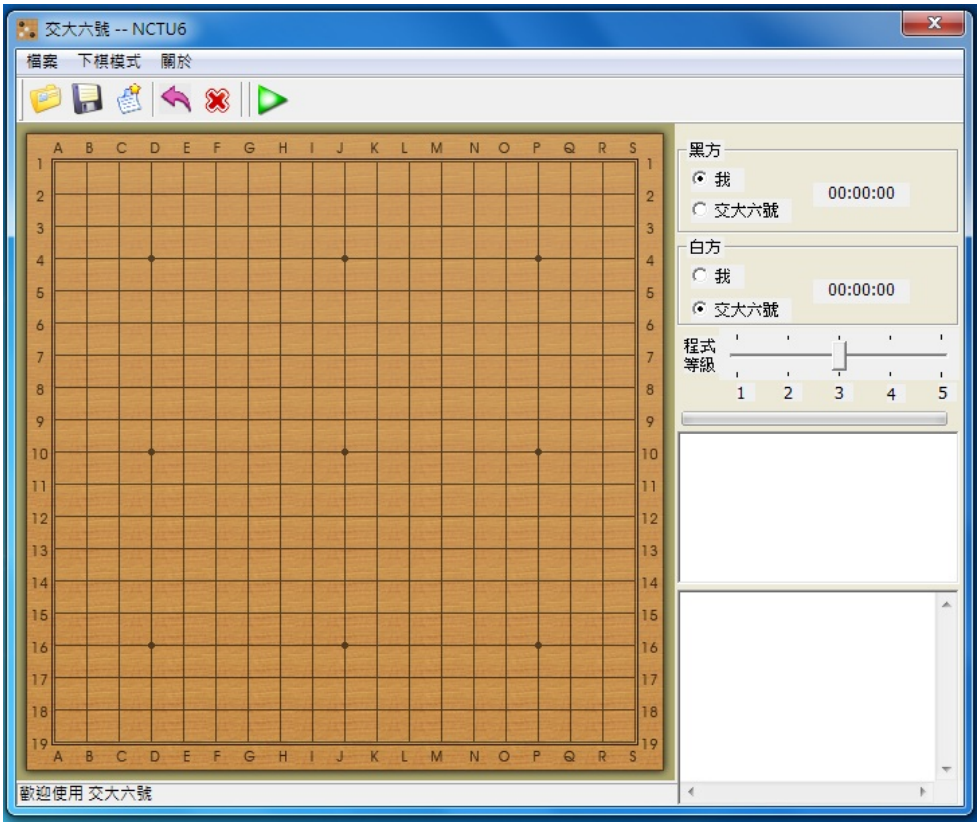
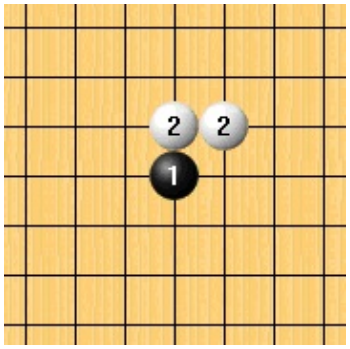
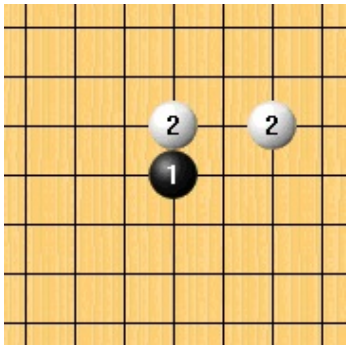


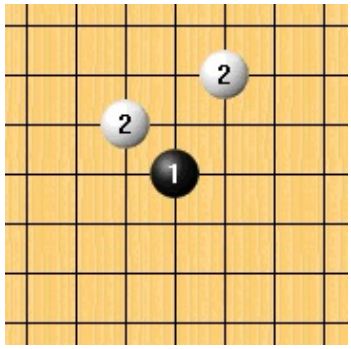
圖 4-3　NCTU6 程式之介面



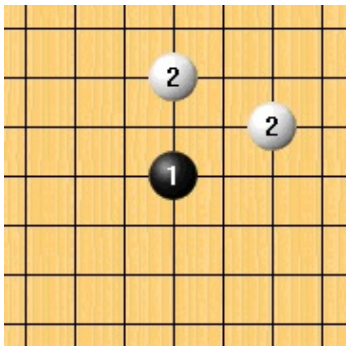
盤面(a)



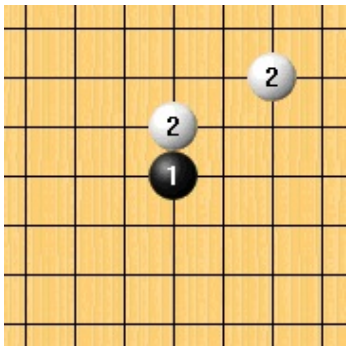
盤面(b)



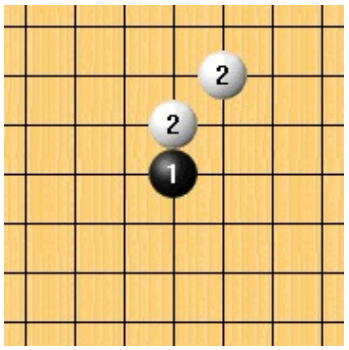
盤面(c)



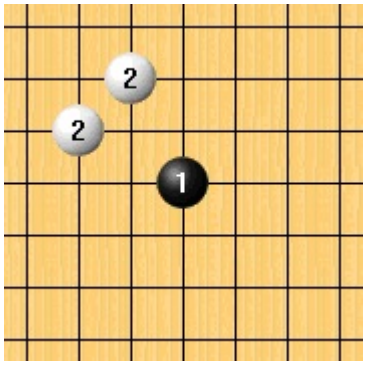
盤面(d)



盤面(e)



盤面(f)



盤面(g)

圖 4-4　本研究之測試盤面

4.2 Ant_1.4 版本測試結果

Ant_1.4 版本由賴昱臣所研發，是未經過本論文作者修改過的版本，本節使用上述的 7 個測試盤面，先與 X6、NCTU6 程式對戰，以此對戰結果做為日後比較的數據。表 4-1 列出 Ant_1.4 版本的對戰結果。由於 Ant_1.4 版本與“結合單迫著與雙迫著搜尋之六子棋程式之研發[10]”中所提到的防守型版本相同，故與 X6

程式的對戰將不再重複，只針對 NCTU6 程式進行測試，完整測試盤面收錄於附錄 B。

表 4-1　Ant_1.4 版本的對戰結果

先手 vs. 後手	Ant_1.4 vs. X6	Ant_1.4 vs. NCTU6
測試盤面（a）	敗	敗
測試盤面（b）	勝	敗
測試盤面（c）	敗	敗
測試盤面（d）	敗	敗
測試盤面（e）	敗	敗
測試盤面（f）	敗	敗
測試盤面（g）	敗	敗
勝率	14%	0%
和率	0%	0%
敗率	86%	100%

4.3 Ant_1.6 版本測試結果

Ant_1.6 版本完成第三章提到的棋型分類、審局函數、時間控管之想法，本節使用上述的 7 個測試盤面，將 Ant_1.6 版本跟另外三支六子棋程式對戰，並統計此版本的勝率、和率與敗率。表 4-2 為 Ant_1.6 版本的對戰結果。完整盤面收錄在附錄 C。

表 4-2　Ant_1.6 版本的對戰結果

先手 vs. 後手	Ant_1.4 vs. Ant_1.6	Ant_1.6 vs. X6	Ant_1.6 vs. NCTU6
測試盤面（a）	和	和	和
測試盤面（b）	和	和	和
測試盤面（c）	和	和	敗
測試盤面（d）	勝	和	和
測試盤面（e）	和	和	敗
測試盤面（f）	敗	和	和
測試盤面（g）	勝	和	敗
勝率	29%	0%	0%
和率	57%	100%	57%
敗率	14%	0%	43%

4.4 Ant_1.7 版本測試結果

Ant_1.7 版本為結合第三章提到的棋型分類、審局函數、候選步的篩選、時間控管、迫著搜尋系統，本節將以此版本重新使用 7 個測試盤面，並將與另外三支程式對戰的勝率、和率與敗率一一列出。表 4-3 為 Ant_1.7 版本的對戰結果。完整盤面收錄在附錄 D。

表 4-3　Ant_1.7 版本的對戰結果

先手 vs. 後手	Ant_1.4 vs. Ant_1.7	Ant_1.7 vs. X6	Ant_1.7 vs. NCTU6
測試盤面（a）	和	勝	和
測試盤面（b）	和	和	勝
測試盤面（c）	和	和	敗
測試盤面（d）	勝	和	勝
測試盤面（e）	和	和	和
測試盤面（f）	勝	和	勝
測試盤面（g）	勝	和	和
勝率	43%	14%	43%
和率	57%	86%	43%
敗率	0%	0%	14%

4.5 各版本 Ant 程式之比較

本節將針對各版本的 Ant 程式，做一完整的比較，用來檢視各種研究方法對程式棋力的改進，並以圖、表方式呈現。

4.5.1　Ant_1.4 版本 vs. Ant_1.6 版本

這裡利用 X6 與 NCTU6 程式作為比較基礎，以檢視棋型分類、審局函數及時間控管對程式棋力的影響。實現上述三種想法後，雖勝率有些許下滑，但和率有很明顯的提升，敗率顯著的降低。表 4-4 列出 Ant_1.4 版本與 Ant_1.6 版本以 X6 程式為對手的勝負比。圖 4-5 繪出 Ant_1.4 版本與 Ant_1.6 版本以 X6 程式為對手的比較圖。表 4-5 是以 NCTU6 程式為對手，統計 Ant_1.4 與 Ant_1.6 兩版本的勝負比。圖 4-6 則以 NCTU6 程式為對手，繪出 Ant_1.4 版本及 Ant_1.6 版本之比較。

表 4-4　以 X6 為對手，Ant_1.4 與 Ant_1.6 之比較

	勝率	和率	敗率
Ant_1.4 版本	14%	0%	86%
Ant_1.6 版本	0%	100%	0%

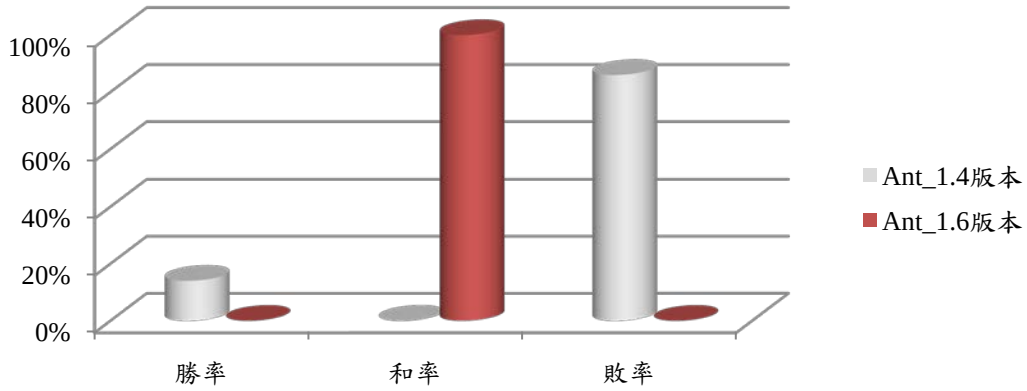


圖 4-5　以 X6 為對手，Ant_1.4 與 Ant_1.6 之比較

表 4-5　以 NCTU6 為對手，Ant_1.4 與 Ant_1.6 之比較

	勝率	和率	敗率
Ant_1.4 版本	0%	0%	100%
Ant_1.6 版本	0%	57%	43%

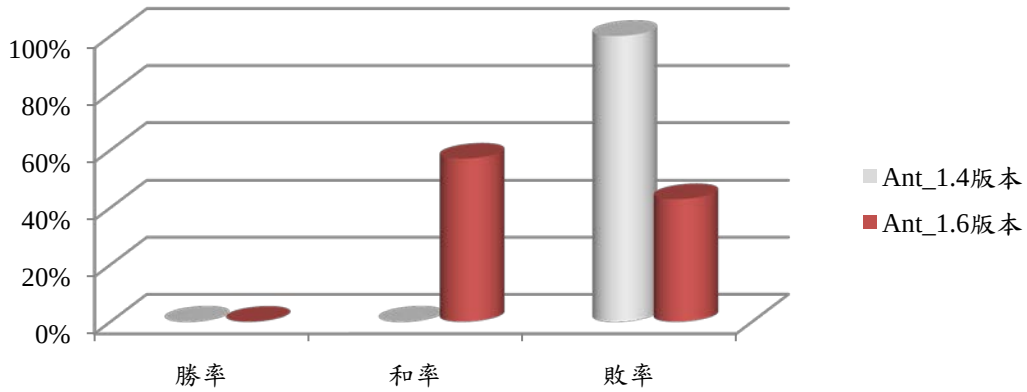


圖 4-6　以 NCTU6 為對手，Ant_1.4 與 Ant_1.6 之比較

4.5.2 Ant_1.4 版本 vs. Ant_1.7 版本

前面已顯示棋型分類、審局函數、時間控管對棋力的重要性，接下來再加上候選步的篩選與迫著搜尋系統，不但勝率及和率都提升，敗率也很明顯下降。表 4-6、4-7 分別以 X6、NCTU6 程式為對手，列出 Ant_1.4 版本與 Ant_1.7 版本的實驗結果。圖 4-7、4-8 分別以 X6、NCTU6 程式為對手，呈現 Ant_1.4 版本與 Ant_1.7 版本的比較圖。

表 4-6　以 X6 為對手，Ant_1.4 與 Ant_1.7 之比較

	勝率	和率	敗率
Ant_1.4 版本	14%	0%	86%
Ant_1.7 版本	14%	86%	0%

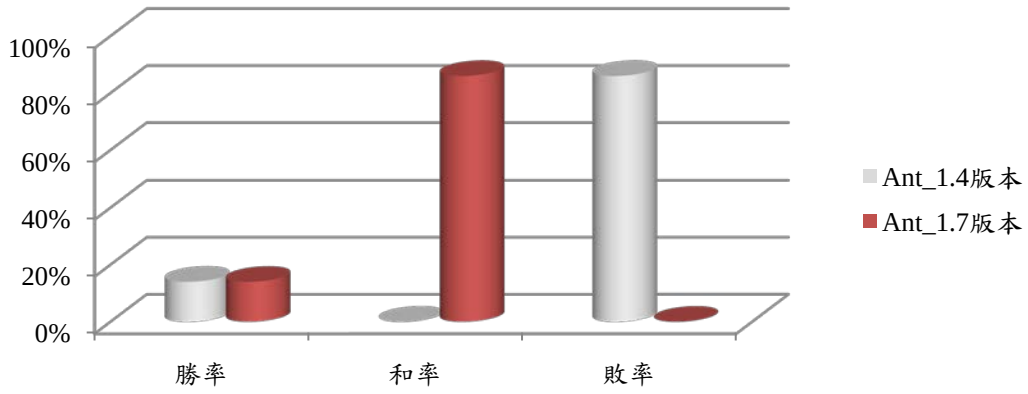


圖 4-7　以 X6 為對手，Ant_1.4 與 Ant_1.7 之比較

表 4-7　以 NCTU6 為對手，Ant_1.4 與 Ant_1.7 之比較

	勝率	和率	敗率
Ant_1.4 版本	0%	0%	100%
Ant_1.7 版本	43%	43%	14%

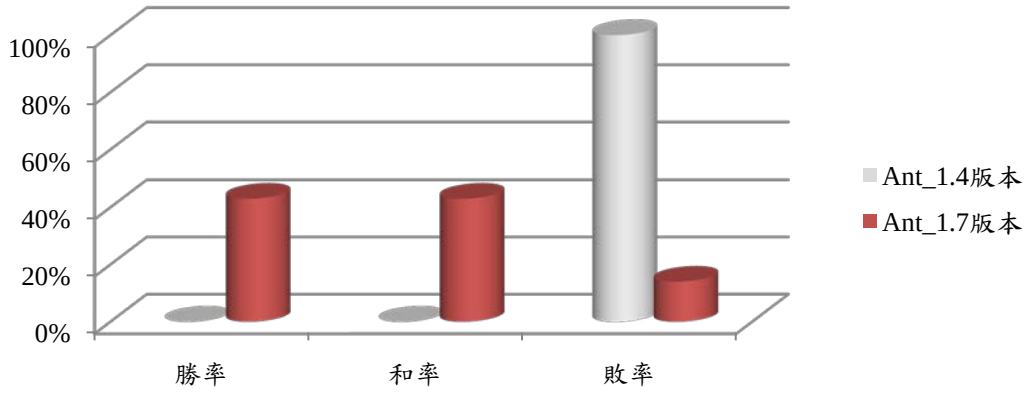


圖 4-8　以 NCTU6 為對手，Ant_1.4 與 Ant_1.7 之比較

4.5.3　Ant_1.6 版本 vs. Ant_1.7 版本

由測試結果可以看出，在對戰時替對手做迫著搜尋是十分有幫助的，另外，候選步的篩選亦可節省部分計算時間，而這些省下來的時間也可讓搜尋展開更多節點。以這些測試盤面來看，勝率明顯比未替對手迫著搜尋高出許多。表 4-8、4-9、4-10 分別以 Ant_1.4、X6、NCTU6 程式為比較基準，列出 Ant_1.6 版本與 Ant_1.7 版本的實驗結果。圖 4-9、4-10、4-11 分別以 Ant_1.4、X6、NCTU6 程式為比較基準，顯示 Ant_1.6 版本與 Ant_1.7 版本的比較圖

表 4-8　以 Ant_1.4 為對手，Ant_1.6 與 Ant_1.7 之比較

	勝率	和率	敗率
Ant_1.6 版本	29%	57%	14%
Ant_1.7 版本	43%	57%	0%

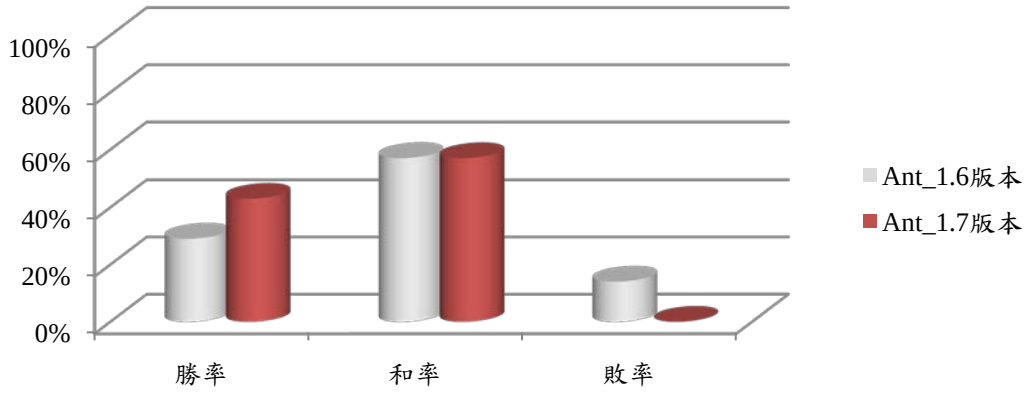


圖 4-9　以 Ant_1.4 為對手，Ant_1.6 與 Ant_1.7 之比較

表 4-9　以 X6 為對手，Ant_1.6 與 Ant_1.7 之比較

	勝率	和率	敗率
Ant_1.6 版本	0%	100%	0%
Ant_1.7 版本	14%	86%	0%

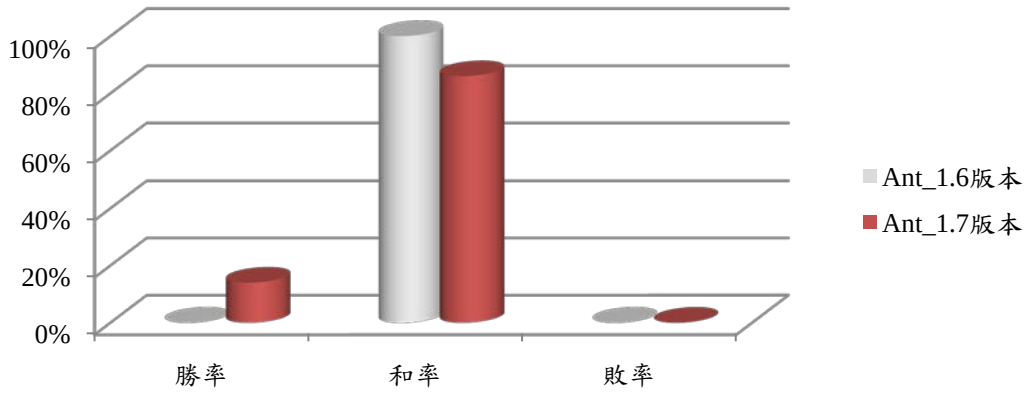


圖 4-10　以 X6 為對手，Ant_1.6 與 Ant_1.7 之比較

表 4-10　以 NCTU6 為對手，Ant_1.6 與 Ant_1.7 之比較

	勝率	和率	敗率
Ant_1.6 版本	0%	57%	43%
Ant_1.7 版本	43%	43%	14%

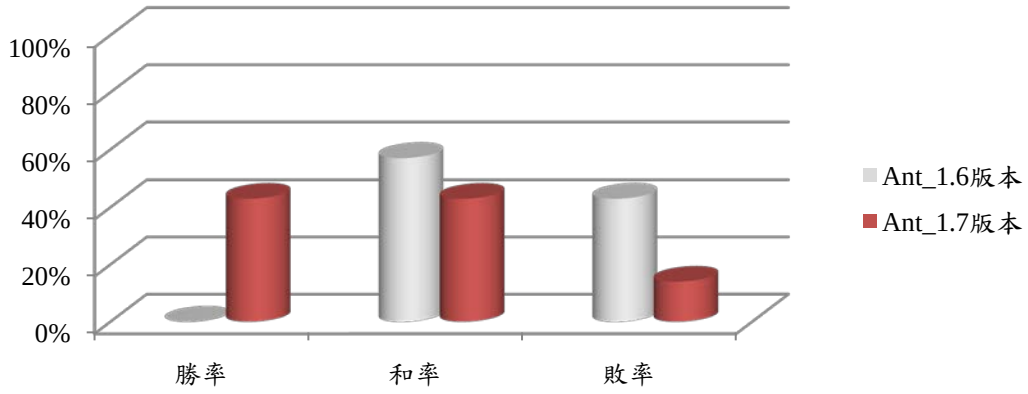


圖 4-11　以 NCTU6 為對手，Ant_1.6 與 Ant_1.7 之比較

第5章　結論與分析

經過實驗測試,發現本研究所提出的棋型分類、審局函數與時間控管等方法，確實可改進程式的棋力，雖然勝率有些許下滑，但和率與敗率皆獲得大幅度的改善，尤其是與 X6 程式對戰，效果更是明顯。候選步的篩選，也讓程式減少許多不必要的計算，使得後續盤面有更充裕的時間思考。而提升棋力最重要的一環為迫著搜尋系統的實現，在我方無法迫著成功的情形下，如何準確抵擋對方的攻勢就變得十分重要，此時搭配替對手迫著搜尋的概念，讓程式可以找出對方接下來的佈局，有效化解對手的攻勢，將原本劣勢的盤面轉成均勢，如此一來才有機會在往後的盤面繼續組織攻勢，進而將均勢化為優勢。從實驗數據顯示，替對手迫著搜尋的想法，也是提高程式勝率、降低敗率的關鍵點。

雖然第三章所提的方法可大幅提升棋力，但在與 NCTU6 的對戰當中，仍會有一些敗陣的狀況，這也暗示 Ant_1.7 版本在審局方面尚不夠精準、搜尋方面不夠全面。另外，在迫著搜尋方面，由於單迫著搜尋的誤判率較高，如何降低誤判風險亦是非常重要的研究方向之一。其他如最近被廣泛運用在棋類遊戲的 Monte-Carlo simulation 算法，或者 Volunteer computing，也是十分值得在六子棋上探討的議題。

參考文獻

[1] Allis, L.V., “Searching for Solutions in Games and Artificial Intelligence,” Ph.D. Thesis, University of Limburg, Maastricht, 1994.

[2] Herik, H. Jaap van den, Uiterwijk, Jos W. H. M., and Rijswijck, Jack van, “ Games solved: Now and in the future,” *Artificial Intelligence*, vol. 134, pp. 277-311, 2002.

[3] Homepage for Connect6, http://www.connect6.org/web/index.php?lang=tw.

[4] Introduction for Theo van der Storm, http://chessprogramming.wikispaces.com/Theo+van+der+Storm.

[5] The results of ICGA 2010, http://ticc.uvt.nl/icga/cg2010results/Connect6.html.

[6] Tournament results of TAAI 2010, http://taai2010.nctu.edu.tw/index.php?option=com_content&view=article&id=71&Itemid=91.

[7] Wu, I-C. and Huang, D.-Y. “A New Family of k-in-a-row Games, ” *ACG11*, September 2005.

[8] Wu, I-C., Huang, D.-Y., and Chang, H.-C. “Connect6, ” *ICGA Journal*, vol. 28, No. 4, pp. 235-242, December 2005.

[9] 劉思源，電腦六子棋程式 X6 之設計與實作。國立東華大學碩士論文，2006 年。

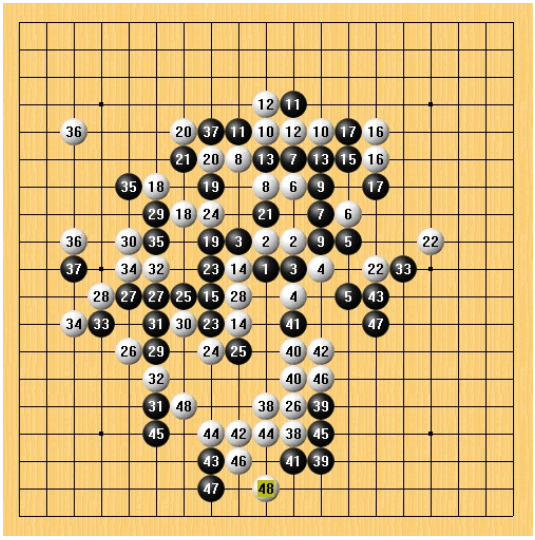
[10] 賴昱臣，結合單迫著與雙迫著搜尋之六子棋程式之研發。國立臺灣師範大學碩士論文，2010 年。

附錄A　TAAI 2010 獎牌照片

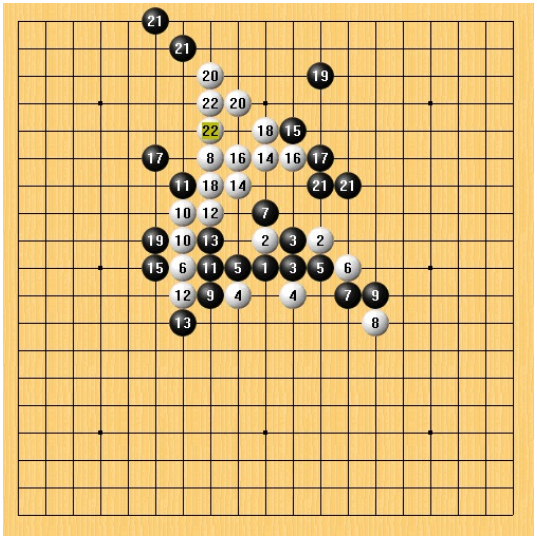
銀牌正面

銀牌背面

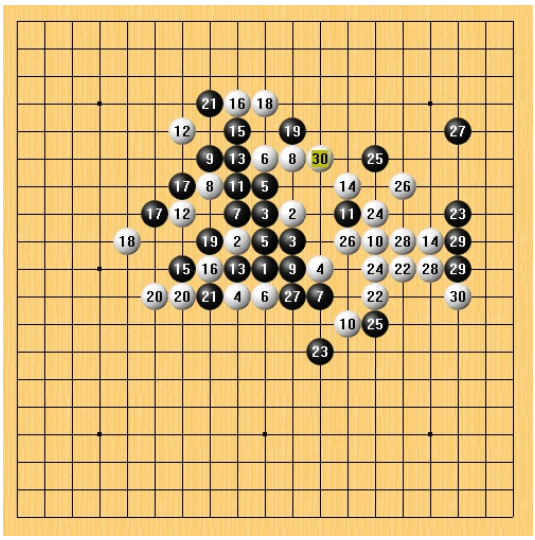
附錄B　Ant_1.4 版本之測試盤面



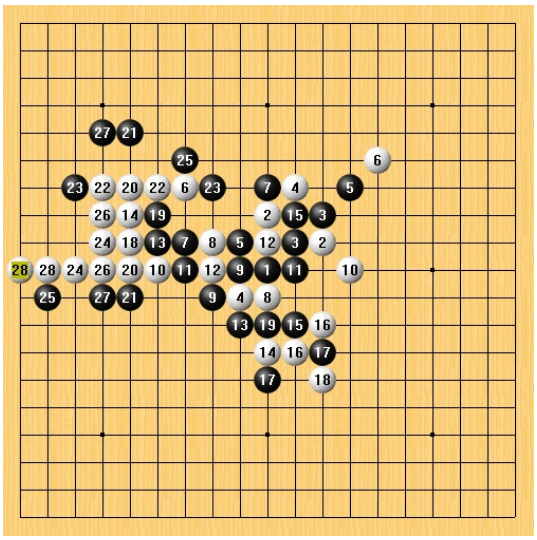
盤面(a)　Ant_1.4 vs. NCTU6



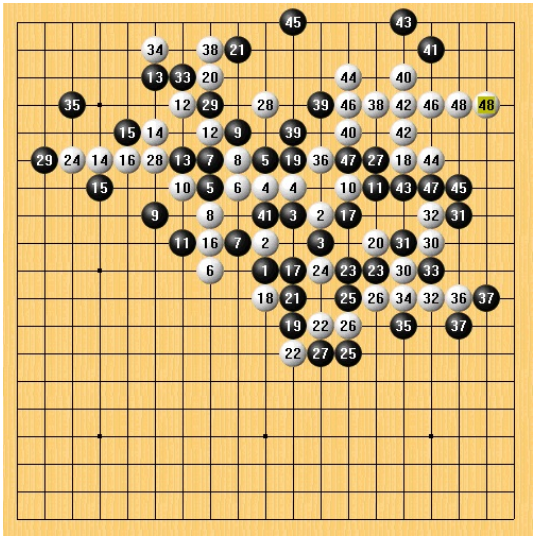
盤面(b)　Ant_1.4 vs. NCTU6



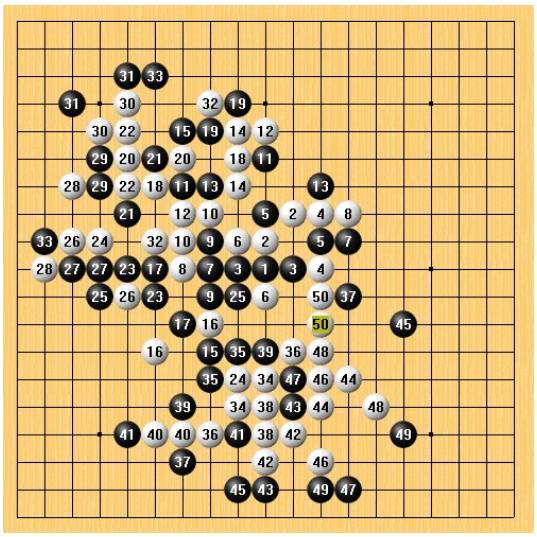
盤面(c)　Ant_1.4 vs. NCTU6



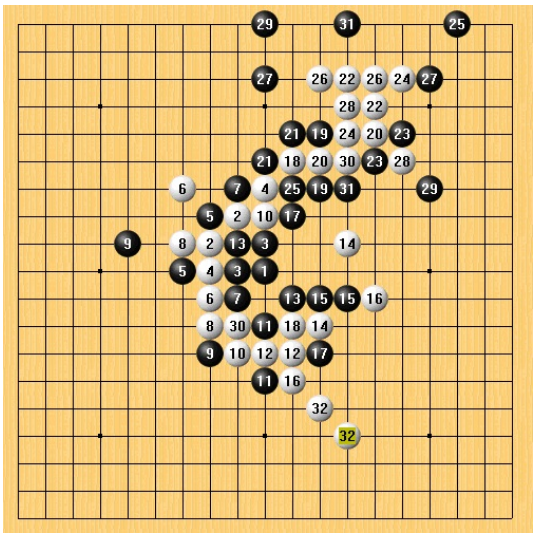
盤面(d)　Ant_1.4 vs. NCTU6



盤面(e)　Ant_1.4 vs. NCTU6

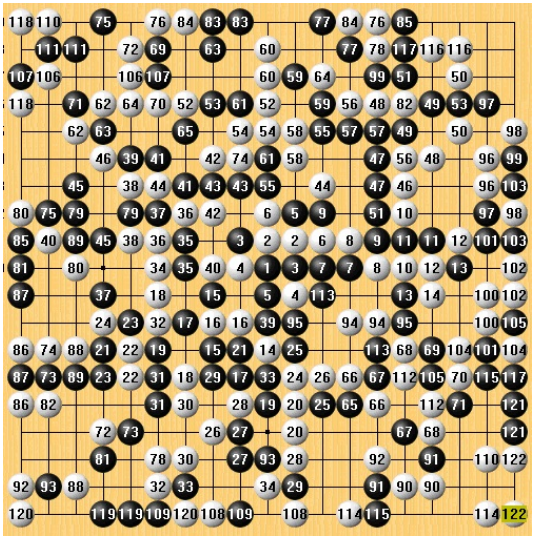


盤面(f)　Ant_1.4 vs. NCTU6

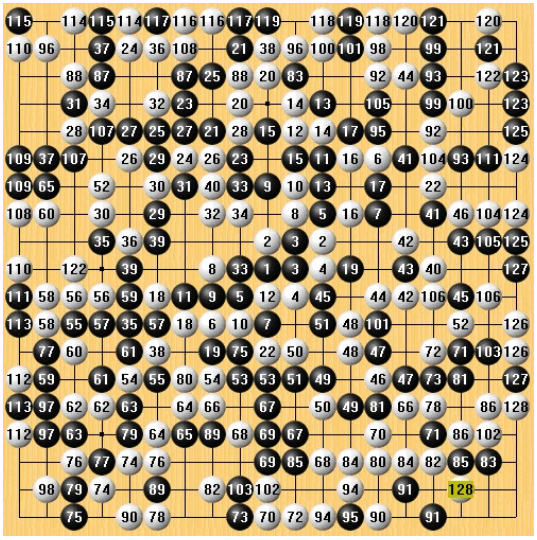


盤面(g)　Ant_1.4 vs. NCTU6

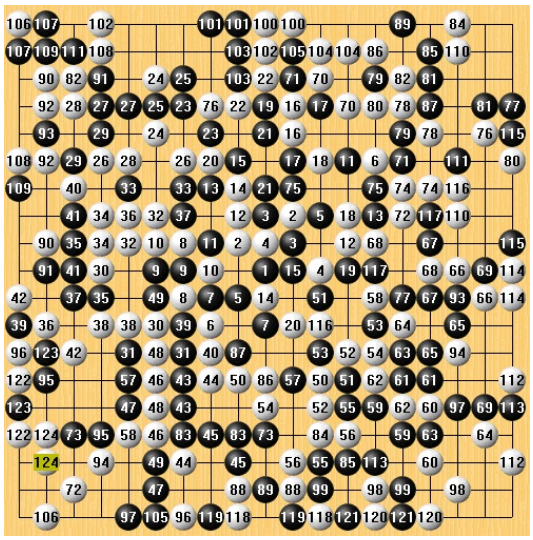
附錄C　Ant_1.6 版本之測試盤面



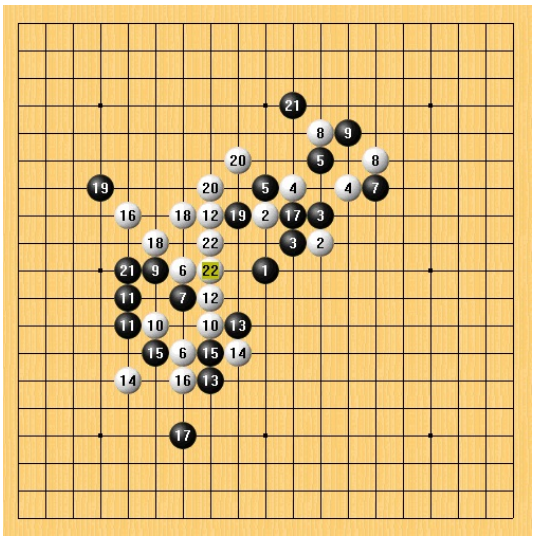
盤面(a)　Ant_1.4 vs. Ant_1.6



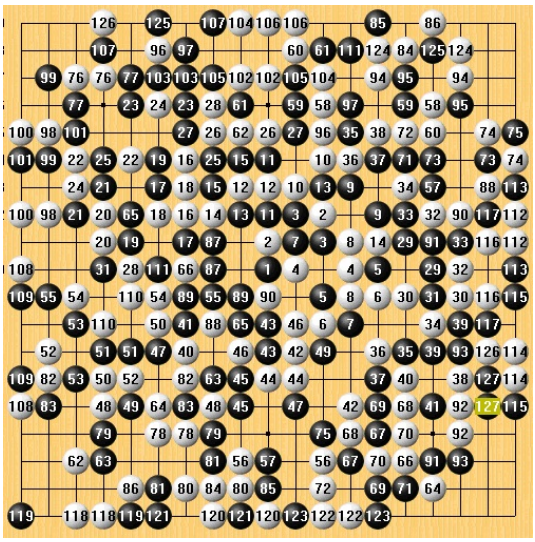
盤面(b)　Ant_1.4 vs. Ant_1.6



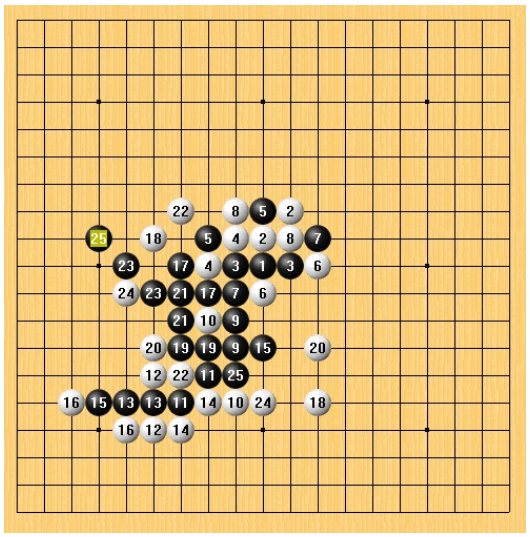
盤面(c)　Ant_1.4 vs. Ant_1.6



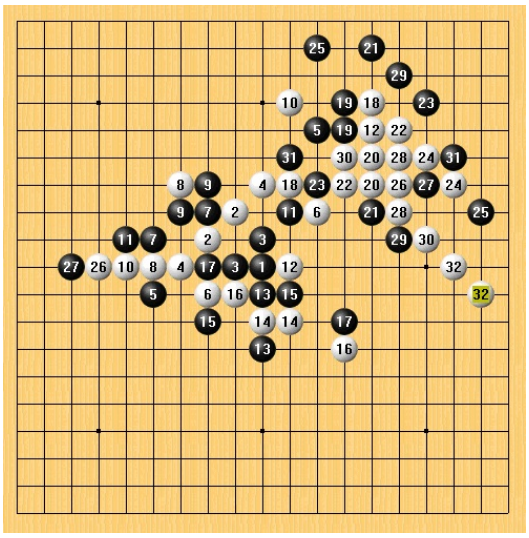
盤面(d)　Ant_1.4 vs. Ant_1.6



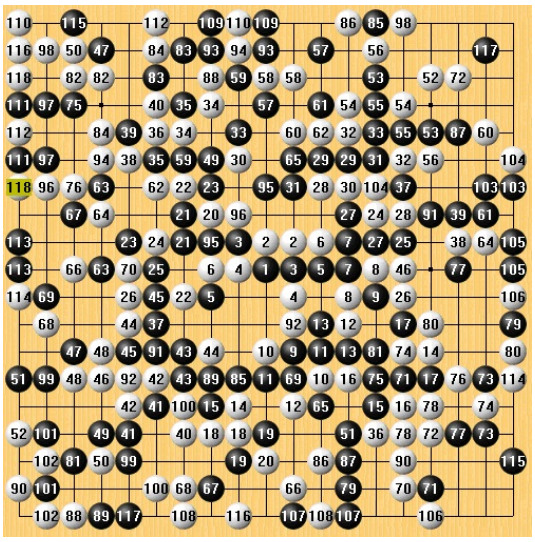
盤面(e)　Ant_1.4 vs. Ant_1.6



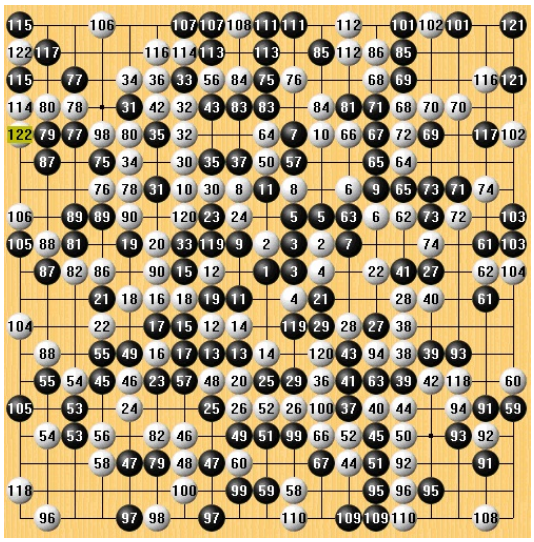
盤面(f)　Ant_1.4 vs. Ant_1.6



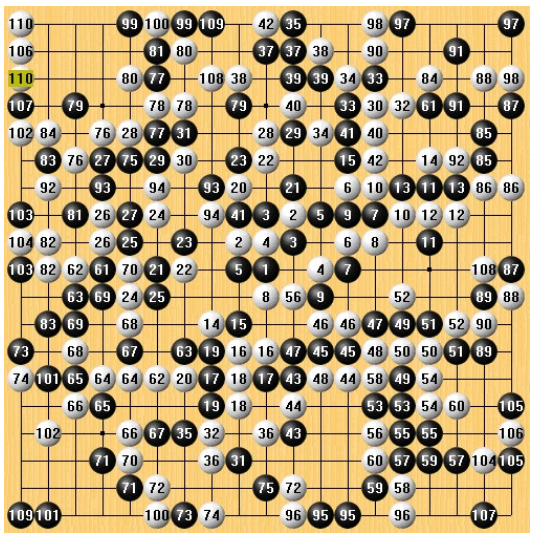
盤面(g)　Ant_1.4 vs. Ant_1.6



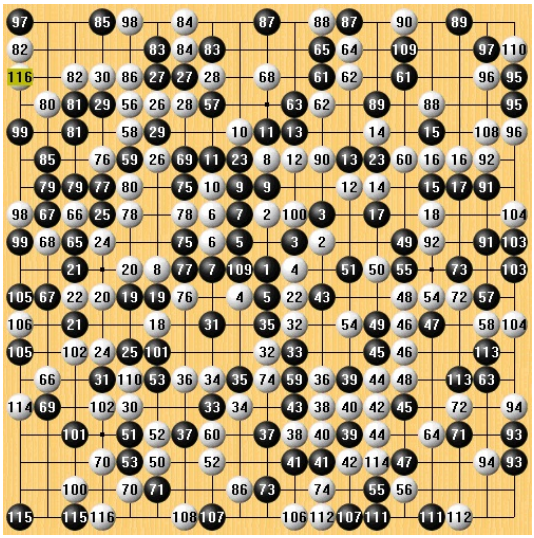
盤面(a)　Ant_1.6 vs. X6



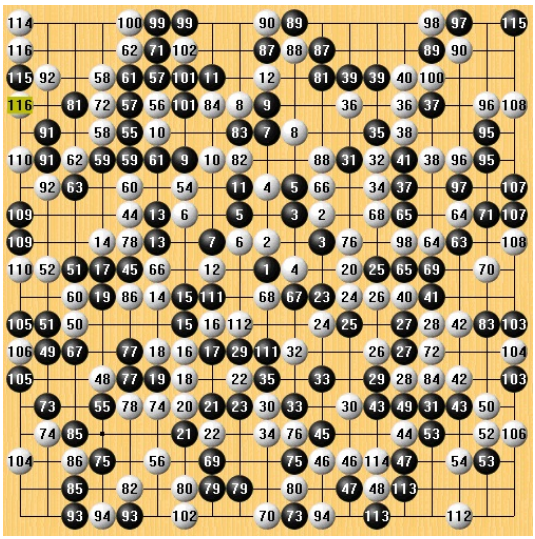
盤面(b)　Ant_1.6 vs. X6



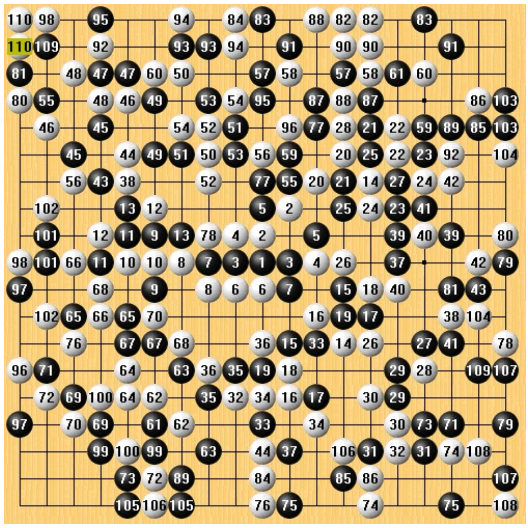
盤面(c)　Ant_1.6 vs. X6



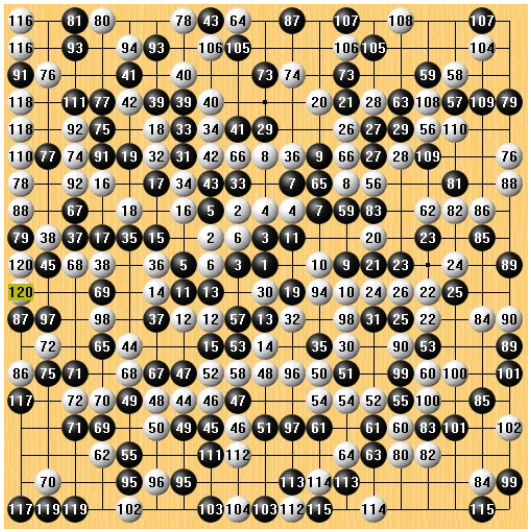
盤面(d)　Ant_1.6 vs. X6



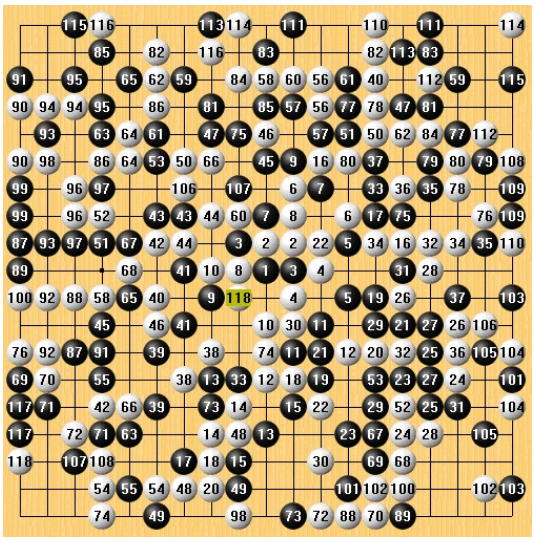
盤面(e)　Ant_1.6 vs. X6



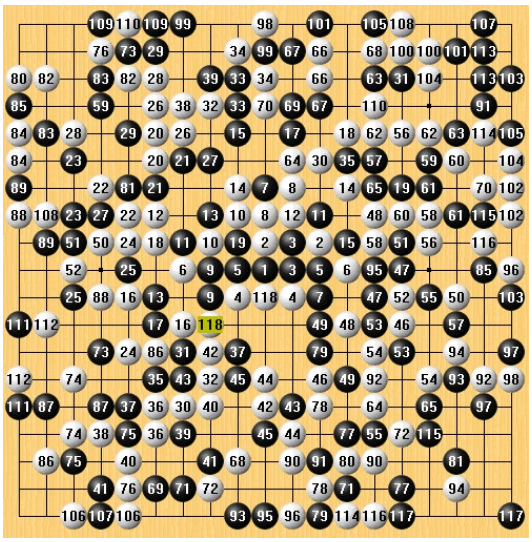
盤面(f)　Ant_1.6 vs. X6



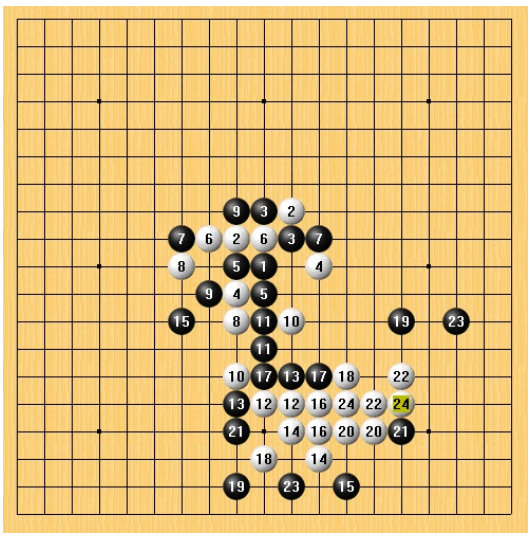
盤面(g)　Ant_1.6 vs. X6



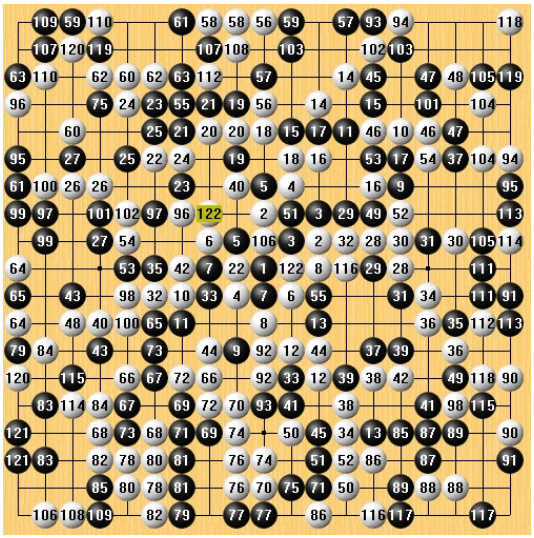
盤面(a)　Ant_1.6 vs. NCTU6



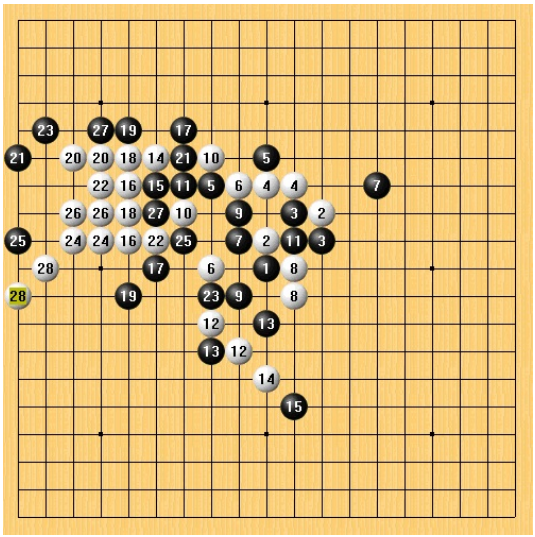
盤面(b)　Ant_1.6 vs. NCTU6



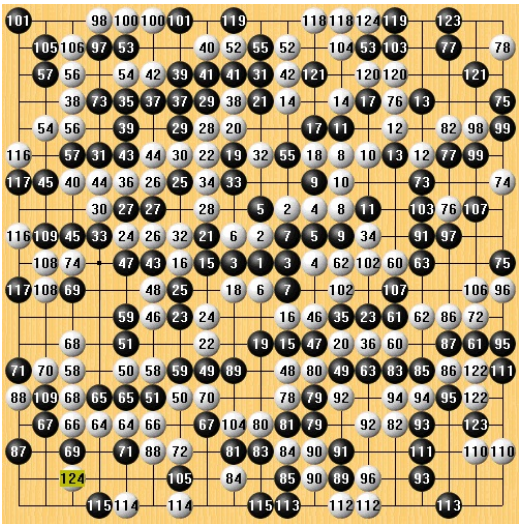
盤面(c)　Ant_1.6 vs. NCTU6



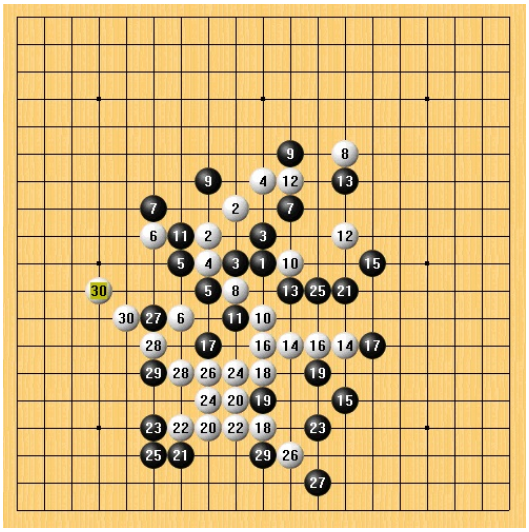
盤面(d)　Ant_1.6 vs. NCTU6



盤面(e)　Ant_1.6 vs. NCTU6

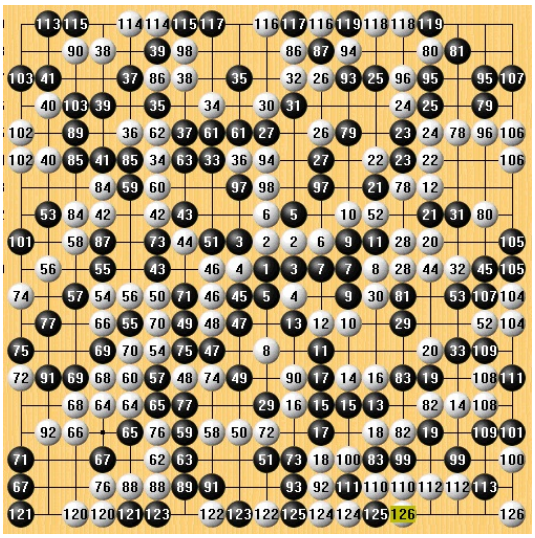


盤面(f)　Ant_1.6 vs. NCTU6

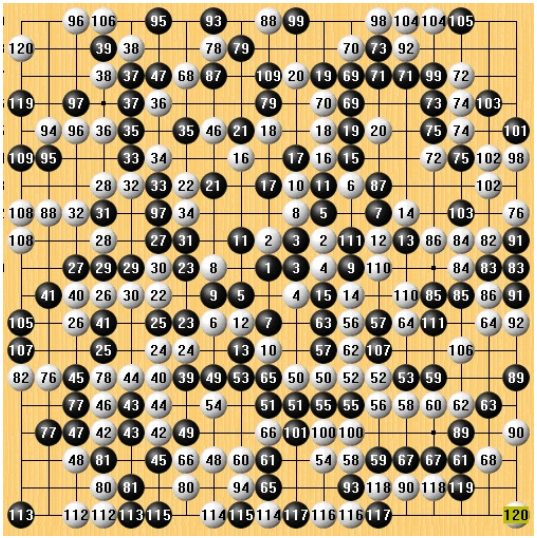


盤面(g)　Ant_1.6 vs. NCTU6

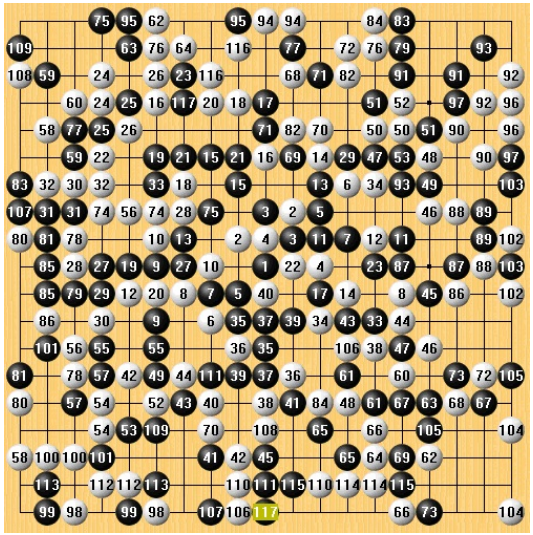
附錄D　Ant_1.7 版本之測試盤面



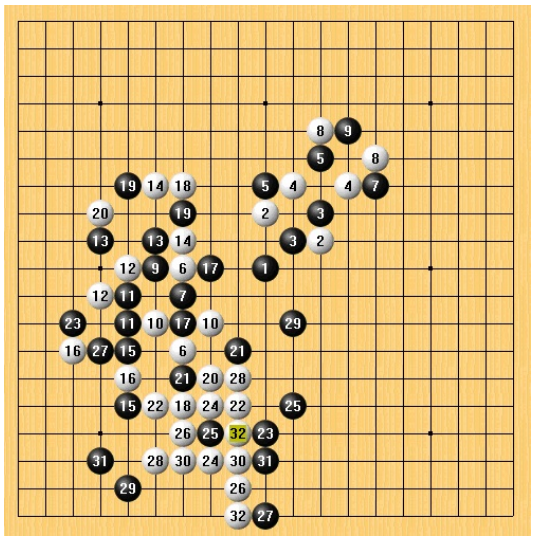
盤面(a)　Ant_1.4 vs. Ant_1.7



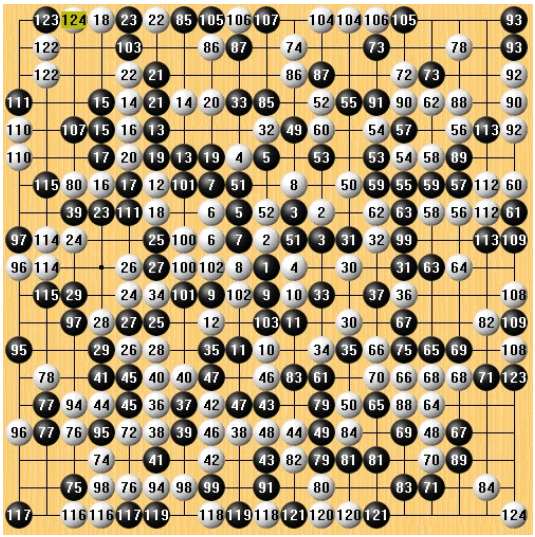
盤面(b)　Ant_1.4 vs. Ant_1.7



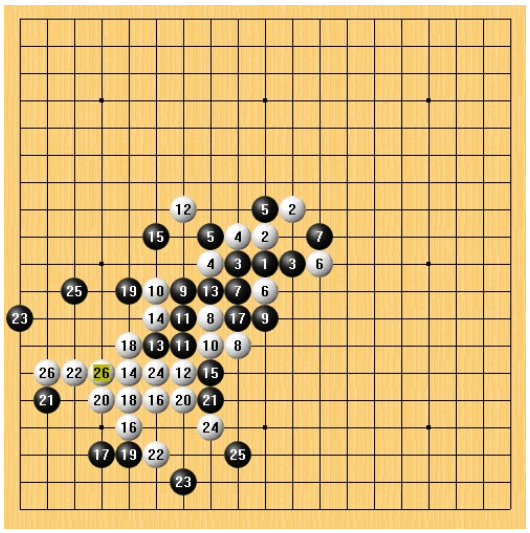
盤面(c)　Ant_1.4 vs. Ant_1.7



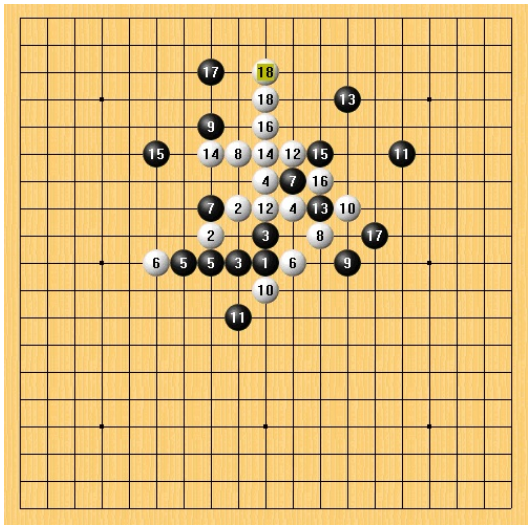
盤面(d)　Ant_1.4 vs. Ant_1.7



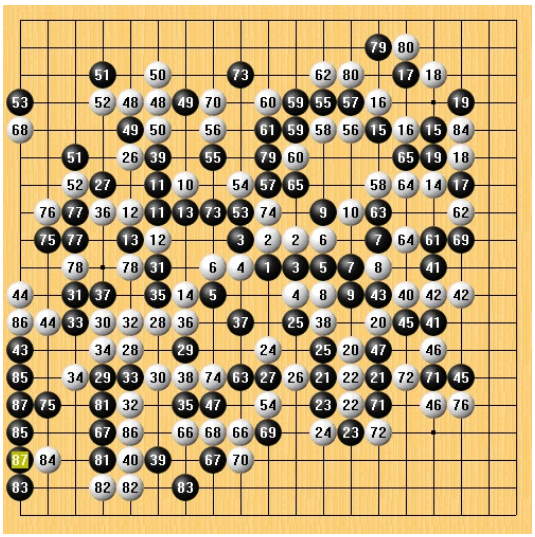
盤面(e)　Ant_1.4 vs. Ant_1.7



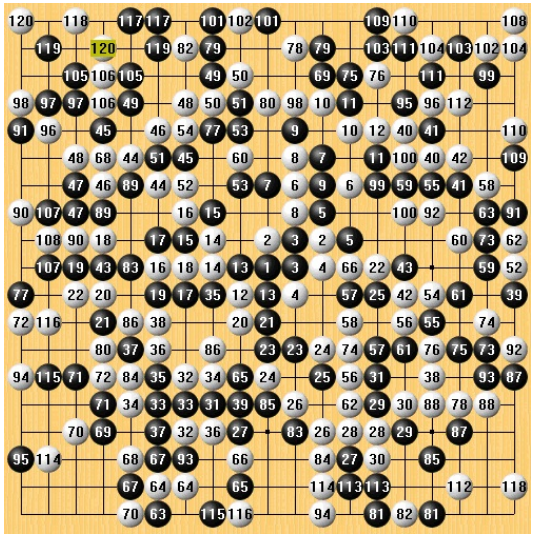
盤面(f)　Ant_1.4 vs. Ant_1.7



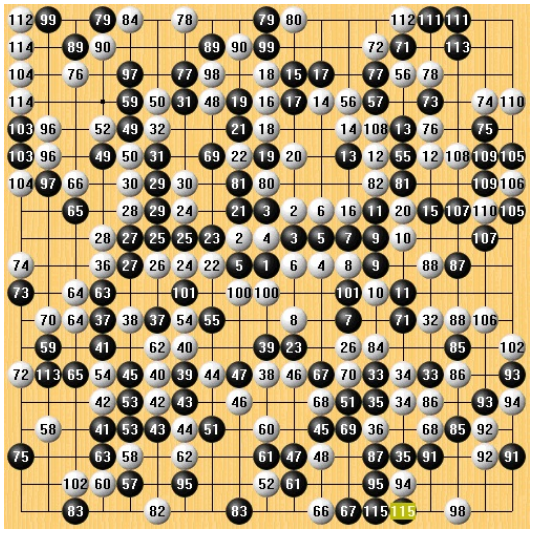
盤面(g)　Ant_1.4 vs. Ant_1.7



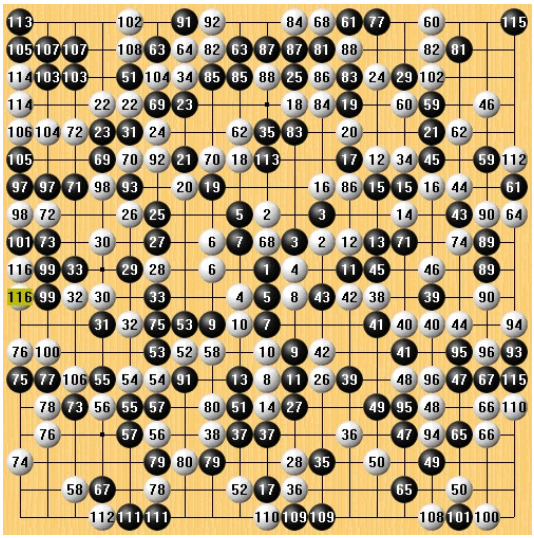
盤面(a)　Ant_1.7 vs. X6



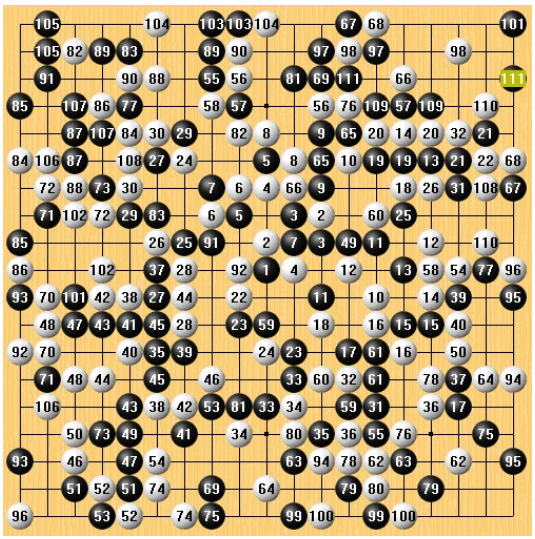
盤面(b)　Ant_1.7 vs. X6



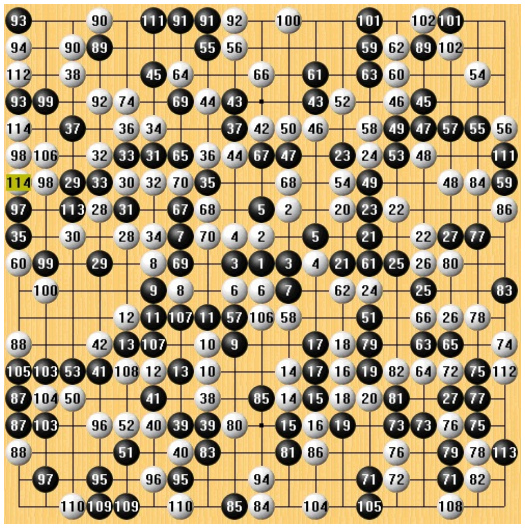
盤面(c)　Ant_1.7 vs. X6



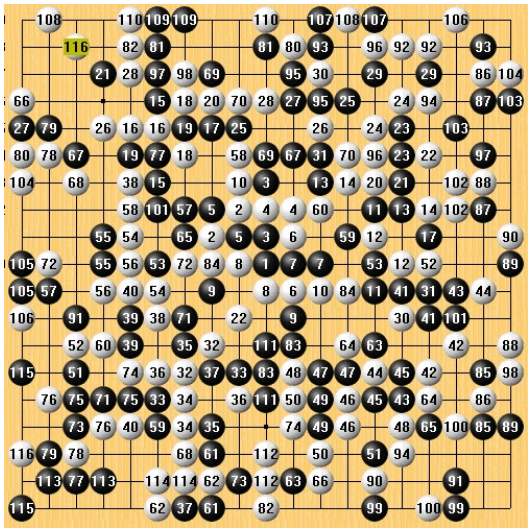
盤面(d)　Ant_1.7 vs. X6



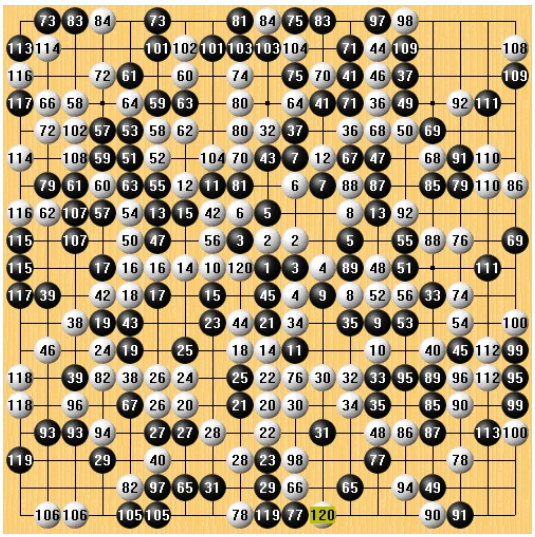
盤面(e)　Ant_1.7 vs. X6



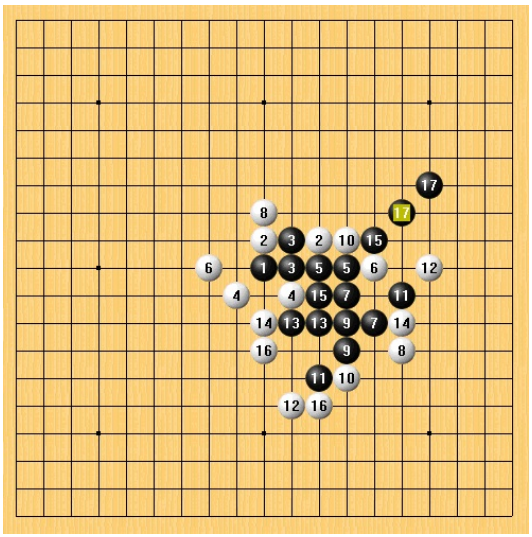
盤面(f)　Ant_1.7 vs. X6



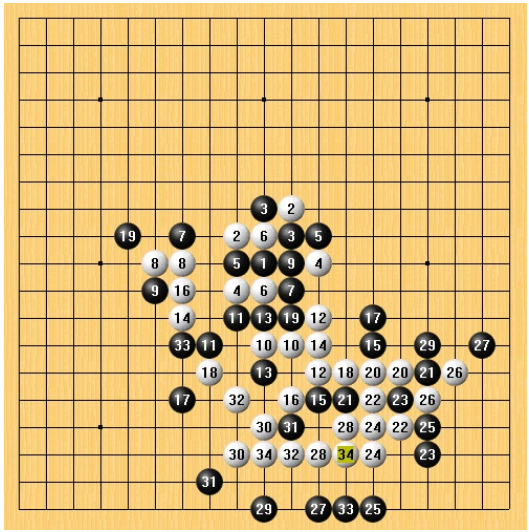
盤面(g)　Ant_1.7 vs. X6



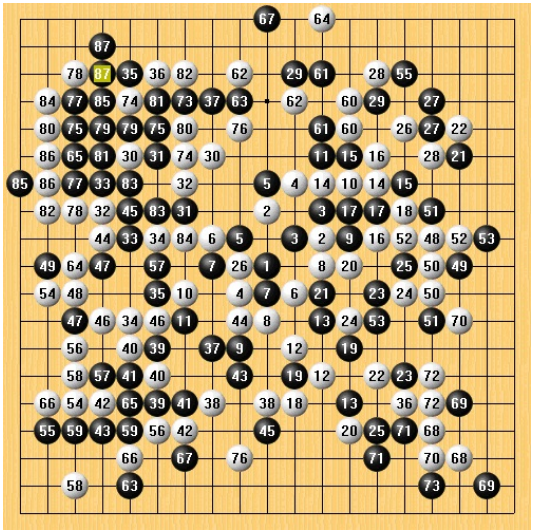
盤面(a)　Ant_1.7 vs. NCTU6



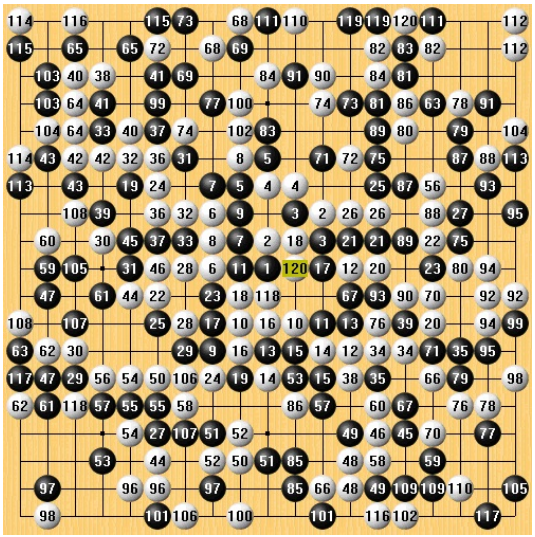
盤面(b)　Ant_1.7 vs. NCTU6



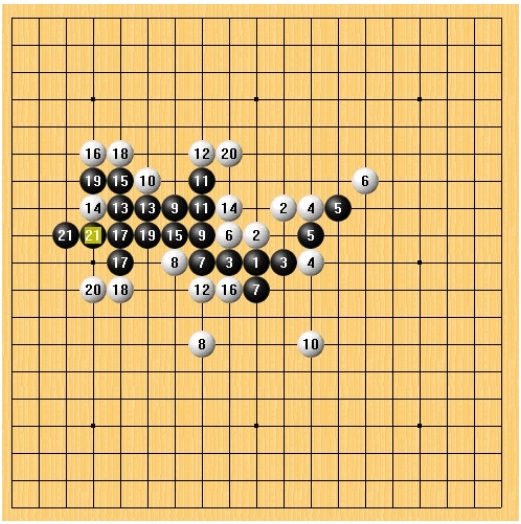
盤面(c)　Ant_1.7 vs. NCTU6



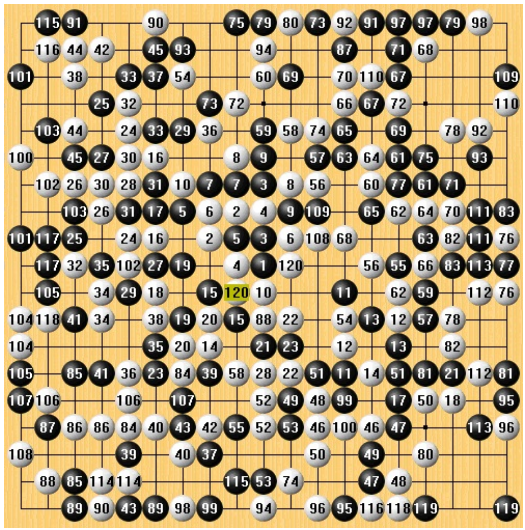
盤面(d)　Ant_1.7 vs. NCTU6



盤面(e)　Ant_1.7 vs. NCTU6



盤面(f)　Ant_1.7 vs. NCTU6



盤面(g)　Ant_1.7 vs. NCTU6